Collaborative Feature Modeling: A Voting Based Approach with Divergence Tolerance and Consensus Facilitation

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

Xin Zhou
China Research Laboratory
IBM
Beijing, China
zhouxin@cn.ibm.com

Abstract—**Feature models provide an effective way to organize and reuse requirements in specific software domains. Many feature modeling approaches have been proposed to guide the construction of feature models, and in most of these approaches, collaboration among stakeholders is considered to be one of the key factors to the quality of constructed feature models. However, few of these approaches provide explicit mechanisms or methods to support such collaboration. In this paper, we propose a collaborative feature modeling approach, an approach that aims to explicitly support collaboration among stakeholders. The basic idea of this approach is to introduce the voting operations in a feature model's construction - that is, besides adding new elements to a feature model as normal, stakeholders can now vote yes or no on any existing element in the feature model to express their viewpoints to this element. The main characteristic of this approach is that it not only tolerates conflicting viewpoints from different stakeholders, but also facilitates the emergence of consensus among them. A case study on the video playing software domain is introduced to show the feasibility of our approach.**

Keywords-feature model; collaboration; domain analysis

I. INTRODUCTION

In software reuse, feature models provide an effective way to organize and reuse requirements in specific software domains. The concept of feature models was first introduced in the FODA method [14]. 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. To take full advantage of feature models in software reuse, a basic problem is how to construct high quality feature models that capture sufficient commonality and variability requirements in specific software domains.

Many feature modeling approaches have been proposed to guide the construction of feature models, and in most of these approaches, collaboration among stakeholders is considered to be one of the key factors to the quality of constructed feature models. For example, in FODA [14] and FORM [15] methods, during a feature model's construction, domain analysts (i.e. constructors of feature models) should make intensive communications with users and domain experts to gather the necessary information about the domain under consideration and resolve possible conflicts among these stakeholders, and after the feature model is constructed, the model then should be reviewed by domain experts, users, requirements analysts and other related stakeholders, to ensure the quality of the feature model. In FeatuRSEB [11], domain experts are important information sources for domain analysts in the whole domain analysis cycle – context modeling and scoping, domain use case modeling, and domain feature modeling, The reason for collaboration is obvious: most real software domains are complex and often evolve frequently, thus it is usually impossible for only one or a few persons to obtain a comprehensive understanding of a domain without sharing knowledge with others.

However, few of the existing feature modeling approaches integrate explicit mechanisms or methods to support collaboration among stakeholders. One consequence of this problem is that, in feature models' construction, although domain analysts can utilize specialized communication tools (such as emails or online forums) or collaboration tools (such as gIBIS [3] or Wikipedia) to enhance collaboration with other stakeholders, the concepts and process of these tools are totally different from those of feature modeling approaches. Therefore, domain analysts have to switch their work context frequently. On the one hand, they have to conduct knowledge sharing and conflict resolution between stakeholders in these specialized tools; on the other hand, they need to transform acquired knowledge and feedback into feature models. This imposes considerable work load on domain analysts, and makes feature models' construction a time-consuming and error-prone task, and in turn, obstructs constructed feature models from evolving with domains. We believe that the integration of feature modeling approach and collaboration support can improve the situation.

In this paper, we propose a collaborative feature modeling approach, an approach that aims to explicitly support collaboration among stakeholders. The basic idea of this approach is to introduce the voting operations in a feature model's construction - that is, besides adding new elements to a feature model as normal, stakeholders can now vote yes or no on any existing element in the feature model to express their viewpoints to this element. Following this idea, we propose an extended meta-model of feature models to explicitly introduce the voting operations into feature models, and define a process and a set of guidelines to construct feature models in a collaborative way. A tool prototype is also developed to carry out this collaborative feature modeling approach efficiently. To verify the feasibility of this approach, a case study on the video playing software domain is conducted. The results of this case study show that our approach provides a simple but effective way to tolerate conflicting viewpoints from different stakeholders as well as facilitating the emergence of consensus among stakeholders.

The reminder of this paper is organized as follows. Section II introduces some preliminaries of our approach. Section III and IV present the static and dynamic aspects (i.e. the conceptual framework and the modeling process) of our approach, respectively. A case study is introduced in Section V to show the feasibility of our approach, and related work is discussed in Section VI. Finally, Section VII concludes this paper with a brief summary and future work.

II. Preliminaries

In this section, we introduce three core concepts of CSCW systems, and a meta-model of traditional feature models. They serve as bases of the "collaborative" aspect and the "feature modeling" aspect in our approach, respectively.

A. *Core Concepts of CSCW Systems*

The field of Computer Supported Cooperative Work (CSCW) was initiated in the mid-80s with the purpose of studying how technology could support group work [12]. Over the years, CSCW researchers [6][8][9][18][22] have identified several central concepts. Although technologies have changed tremendously, these concepts still shape the design of modern CSCW systems. Among these concepts, the most important three are production, awareness, and coordination.

Production refers to *objects* produced and shared in a CSCW system, and *operations* applied on these objects [18]. It can be represented as the conceptual framework [9] of a CSCW system. The implementation of this concept in our approach will be described in Section III.A and III.B.

Awareness means that every CSCW system should provide necessary information to show the context of individual's work [6]. 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 our approach will be described in Section III.C.

Coordination relates to mediating and meshing individual work in a shared workspace [22]. While "production" defines activities available to collaborators, the term "coordination" focuses on the organization of these activities between them [9] in order to reduce possibilities of conflict or repetition. The coordination mechanism in our approach will be explained in Section IV.C.

B. *A Meta-Model of Feature Models*

Fig. 1 shows a meta-model of traditional feature models. Generally, a feature model consists of a set of features and relationships between them. There are two types of relationships, namely *refinements* and *constraints*.

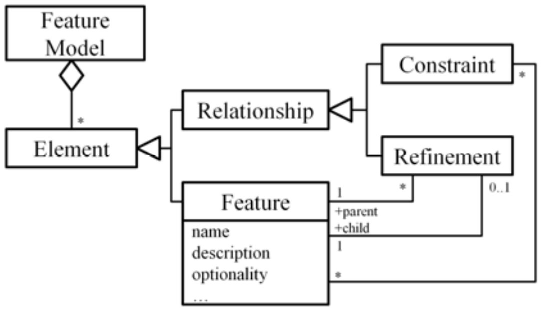


Figure 1. A meta-model of feature models.

The concept of *feature* can be understood in two aspects: intension and extension [25]. In *intension*, a feature denotes a cohesive set of individual requirements. In *extension*, a feature describes a software characteristic having sufficient user/customer value.

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.

The *constraint* relationships describe dependencies between features. There are two basic kinds of constraints: *requires* and *excludes* [14][25]. 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 [25].

III. The Conceptual Framework

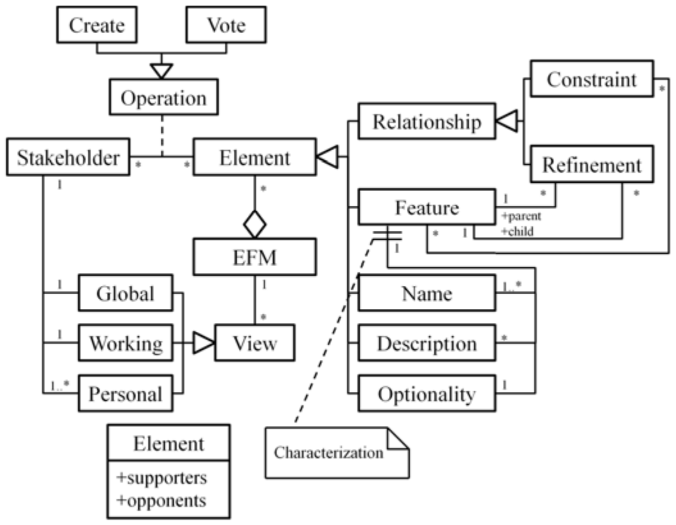
In this section, we explain the extended feature model (EFM) which will be collaboratively constructed. We first introduce the meta-model of EFMs as an overview. Then we give more details about the extensions we make.

A. *The Meta-Model of Extended Feature Models (EFMs)*

Fig. 2 shows the meta-model of extended feature models. An EFM is composed of many vote-able *elements*. The top-level elements are *features* and *relationships*. A feature may have one or more *names*, and each name is a vote-able element as well. Similarly, a feature may have many vote-able *descriptions*. However, a feature has exactly one *optionality* attribute which is an element as well. The result of vote on every element is recorded as its *supporters* and *opponents*, both of which are a set of *stakeholders*.

Figure 2. The meta-model of extended feature models.

Stakeholders work on EFMs with the *operations* we provide. They can either *create* new elements in EFMs or *vote* on existing ones. Other ordinary modeling operations, such as modification or deletion, are implemented through the two basic operations. (See Section III-B for details.)



We provide three types of *views* for each EFM, namely global view, working view and personal view. Each stakeholder has one global view, one working view and at least one personal view for each EFM he involved. The views are derived from EFMs automatically. (See Section III-C for details.)

B. *Operations on EFMs*

We provide two operations for stakeholders, namely creating and voting. Beside, there are restrictions on creating operations, and there are Automatic Vote Propagation Rules applying to both operations. Finally, with the help of the rules, stakeholders can simulate deletion, modification and moving on elements.

1) *Creating and restrictions*

The creating operation is less restricted in EFMs than in feature models (FMs) introduced in Section II. For example, stakeholders can create several names for a single feature in EFMs, while they can only give exactly one name for each feature in FMs. In fact, the only restriction on creation is that no duplicate elements can be created (except for Optionality), as described in Table I.

TABLE I. RESTRICTIONS ON CREATION

Element	Restrictions
Name	The name has not been used in any of the features in current EFM.
Description	The description has not been used in current feature.
Feature	A name for the feature must be created successfully before creating the feature.
Refinement (X refines Y)	X and Y are existing features in current EFM and no "X refines Y" existed.
Constraint (X requires Y)	X and Y are existing features in current EFM and no "X requires Y" existed.
Constriant (X excludes Y)	X and Y are existing features in current EFM and no "X excludes Y" or "Y excludes X" existed.
Optionality	Its values are predefined (*mandatory, optional*), thus no creation is allowed.

In EFMs, a feature may have many names because terminologies used by stakeholders may be different. We treat the names of the same feature as aliases of each other, and no duplicate names are allowed in or between features.

Unlike names, descriptions are not identifiers of features, so we only forbid duplicate descriptions in the same feature.

For relationships, we only list refinements and binary constraints in Table I for simplicity. Restrictions on group and complex constraints defined in [25] are similar.

Finally, the optionality of features has predefined values: mandatory or optional. "Create a new optionality" is meaningless and therefore disallowed.

2) *Voting and automatic vote propagation*

Stakeholders vote yes or no on an existing element to express their viewpoints on the element. First of all, we want to describe the semantic of such vote:

- A "Yes" vote shows support for existence of the voted element in current EFM, while "No" shows opposition, (if type of the element is not Optionality).
- A "Yes" vote on the optionality shows support for treating corresponding feature as an optional one, while "No" treats it as a mandatory one.

The semantic of voting on optionality is special because it is creating-forbidden and can be perceived as a Boolean variable.

There are no restrictions when stakeholders are voting on elements. However, semantics of the votes of the same stakeholder may be inconsistent. For example, if there is a feature "A" and a constraint "A requires B", what does it mean if a stakeholder vote no on the feature but vote yes on the constraint? Based on the semantic of votes, the stakeholder supports a constraint which involves a feature considered nonexistent. Therefore the votes are inconsistent.

Our solution to such problems is to propagate the vote to corresponding elements automatically, whenever a voting operation is submitted. First, we want to find out what are the corresponding elements of the being voted element, by taking a closer look at the relations of existence of elements in an EFM. For clarity's sake, we define a predicate *Exist as:*

$$Exist(Element\ E) = \begin{cases} True,\ if\ \ E\ exists\ in\ the\ EFM, \\ False,\ else. \end{cases}$$

Thus, the relations of existence of elements are:

$$Exist(Relationship\ R\ involving\ Feature\ F) \rightarrow Exist(F), \quad (1)$$

and

$$Exist(Attribute\ A\ of\ Feature\ F) \rightarrow Exist(F). \quad (2)$$

According to the semantic of votes, we immediately know that (we will discuss optionality later),

$$Yes(Element\ E\ not\ Optionality) \leftrightarrow Exist(E), \quad (3)$$

where the predicate *Yes* is defined as,

$$Yes(Elemen\ t\ E) = \begin{cases} True,\ if\ \ vote\ yes\ on\ E, \\ False,\ if\ \ vote\ no\ on\ E. \end{cases}$$

Therefore, according to (1), (2) and (3), we know the target elements of vote propagation, for each type of the being voted element, as shown in the following Automatic Vote Propagation Rules (PRs).

- PR-1:

$$\textit{Yes(Relationship R)} \rightarrow \bigwedge_{\textit{F involved in R}} \textit{Yes(Feature F).}$$

- PR-2:

$$\neg \textit{Yes(Feature F)} \rightarrow \bigwedge_{\textit{R involves F}} \neg \textit{Yes(Relationship R).}$$

PR-1 and PR-2 can be easily derived from each other. They ensure consistent votes on features and relationships from a stakeholder. We apply them to the previous example, where a stakeholder wants to vote on feature "A" and constraint "A requires B". If he votes yes to the constraint, a yes vote to both feature A and B will be propagated. If he votes no on feature A, the constraint will be propagated a no vote, while feature B will not be affected.

Similarly, the next pair of rules defines the vote propagation between features and their names/descriptions.

- PR-3:

Yes(Attribute A of Feature F) →Yes(F), where A typeof Name or Description.

- PR-4:

$$\neg \textit{Yes(Attribute A)} \rightarrow \bigwedge_{\textit{A belongs to F}} \neg \textit{Yes(Feature F), where}$$

A typeof Name or Description.

Now we deal with votes on the optionality attribute. According to previous discussions, the semantic of vote on optionality is whether the current feature is mandatory or not, while the semantic of vote on a feature is whether the feature should exist or not. Given these semantics, we can say that *a feature must exist before its optionality can be considered.* Therefore, we define a pair of rules as below:

- PR-5:

Vote on the optionality of feature F→Yes(F).

- PR-6:

¬Yes(Feature F) → Remove the voter's vote on the optionality of F.

The six PRs above handle the propagation between votes. However, there is a connection between creating and voting operations, according to their semantics. The semantic of creating is to make a currently nonexistent element exist in an EFM, while the voting expresses support or opposition of existence of the element. (Because no creation is allowed for the optionality, we omit optionality here.) Thus we can say that if a stakeholder creates an element, he obviously supports for its existence. Therefore we have the following Creating-imposed Vote Propagation Rule (PR-CV).

- PR-CV:

Create element E → Yes(E), where type of E is not Optionality.

All PRs are transitive, although transitions only take effect on creating operations -- no actual transition happens when submitting voting operations, according to the definition of PR-1 to PR-6. For example, when stakeholders create a relationship, PR-CV applies, which in turn, makes PR-1 take effect.

For voting operations, however, it is possible to apply multiple PRs to a single voting operation. For example, PR-2, 4 and 6 take effect when submitting no vote on a feature.

In addition to seven PRs, we have an extra rule to deal with repetitive votes. Since we have not imposed any restriction on the voting operation, a stakeholder may vote on the same element for many times. Furthermore, operations submitted by the stakeholder may cause several propagated votes on the same element as well. Thus it is highly possible that repetitive voting could happen, and we handle them with the following rule.

- Repetitive Vote Rule (RVR):

If there are multiple votes on an element from the same stakeholder, no matter the votes are submitted by voting operations or propagated, only the last vote counts.

With the help of the rules, stakeholders can simulate other modeling operations via the two operations, as described in the next sub-section.

3) Simulating other modeling operations

Most feature modeling approaches provide operations for creating, deleting, modifying and moving elements in feature models. We call these four operations as *ordinary modeling operations*. As mentioned before, we purposely provide only one ordinary modeling operation – the creating – plus the voting operation, to prevent stakeholders from editing and overwriting others' work directly, thus to satisfy the divergence tolerance property.

However, we do allow stakeholders to simulate the three non-provided operations in EFMs, *if and only if they can reach consensus*. Since the modification and moving are combination of creation and deletion, we focus on simulating the deletion operation first.

The basic idea comes from the fact that deletion is opposite to creation. According to the discussion of PR-CV in the previous sub-section, it seems we can apply PR-CV again but from another direction, as shown below. (Again, since there is no deletion for the optionality attribute, we omit optionality here.)

PR-CV Inversed (PR-CVI):

¬Yes(Element E) → Delete(E), where type of E is not Optionality.

PR-CVI means if a stakeholder votes no to an element, his intention is to oppose the existence of the element, which conforms to the semantic of deletion. Moreover, PR-CVI works perfectly with the PR-2, PR-4, and PR-6, that is, when a feature is deleted, its attributes and involved relationships are also deleted. This looks fairly reasonable.

However, in the context of collaborative work, there might be multiple voters on the same element. Therefore, we define that an element will not be deleted, until all votes on it are no, which means consensus has to be achieved on the element, as below:

- Vote-Simulating Deletion:

Delete element E $=_{def}$ *All votes on E are "No".*

It is worth noticing that the PR-CV ensures an initial "yes" vote on every element, from its creator. Therefore an element can be deleted only if its creator permits the deletion by voting no on it. It again reflects the divergence tolerance characteristic of our approach – every viewpoint of the domain should be and can be reserved.

Another property is the deletion does not require all stakeholders vote no on the element, only *all voters on the* *element*. In other words, it only needs consensus between stakeholders who are interested in the element. Therefore it can reduce unnecessary work load on other stakeholders. Furthermore, this makes the simulation of undo possible, if the only stakeholder works on an element is the creator of it.

The last thing about the deletion is the simulation of deleting features. It seems a little bit tricky, since features are not "atomic" elements – they are composed of elements of the type name, description and optionality. Is it possible that delete a feature could destroy others' contributions to its name, description or optionality? For example, what if a being deleted feature contains a name whose supporter is not one of the voters of the feature? In fact, this cannot happen, thanks to the vote propagation rules PR-3 and PR-5. These rules ensure that every creator or supporter of any name and description of a feature, and every voter of its optionality, must be a supporter of the feature as well. Therefore, every contributor of a feature has the chance to express his opinion on the existence of the feature.

As long as we simulate deletion with creating and voting, the modification and moving are trivial. To modify an element (except for the optionality), stakeholders delete it first, and then create a new one of the same type. To move a feature, stakeholders need to modify the refinements which involve the feature as a child.

C. Views for EFMs

The creating and voting operations introduced in the previous sub-section allow different opinions to coexist in the same EFM. Now we want to discuss how to represent the EFM to stakeholders, with the aim of help them understand current context of work.

Our basic idea is a stakeholder's context of work is built on his own viewpoints on an EFM, thus the context is easy to understand for him. Besides, the whole picture of the EFM, and different stakeholders' understanding of the EFM should be provided.

Therefore, we provide three types of EFM views, namely working view, global view and personal view. For each EFM, every stakeholder has exactly one working view as the main view, as well as one global view and several personal views as supplements. All the views are automatically generated.

1) Working view

Working views act as the main working space of stakeholders. We generate the working view of an EFM for a specific stakeholder according to the following rule.

WV(EFM m, Stakeholder s) = {Element e $|$ *e* $\in$ *m* $\wedge$ *s has not voted no on m}*.

Thus, the working context of a stakeholder consists of,

- Elements he supported, either created by others or himself, and
- Elements created by others and he has not shown any opinions.

Since the voting on optionality has different semantic, it always resides in the features in working views.

2) Global view

Global views reveal the whole picture of an EFM. It is composed of all undeleted elements in an EFM.

GV(EFM m, Any stakeholder) = {Element e $|$ *e* $\in$ *m* $\wedge$ *At least one Yes vote on e}*.

The rule PR-CV introduced in previous section attaches an initial Yes vote on elements, thus ensures global views always correctly generated even if the stakeholders have not explicitly voted yes.

3) Personal view

Personal views dedicate to represent stakeholders' own EFMs, by put the element supported by them together.

PV(EFM m, Stakeholder s) = {Element e $|$ *e* $\in$ *m* $\wedge$ *s has voted yes on e}*.

In a word, the private view of a stakeholder consists of elements he supported, either created by him or by other stakeholders.

Stakeholders can switch to personal views of any others, thus get the snapshot of a specific stakeholder's viewpoint.

IV. The Process

In this section, we introduce the process of collaborative feature modeling, based on the conceptual framework discussed before. We first introduce the overview of the process, and then we describe the conflict resolution and the coordination mechanism in detail, respectively. Finally, we give some guidelines for stakeholders.

A. An Overview of the Process

In this sub-section, we first introduce an overview or the process, and then give more details about the activities in it.

Fig. 6 shows the process of collaborative construction of an EFM.

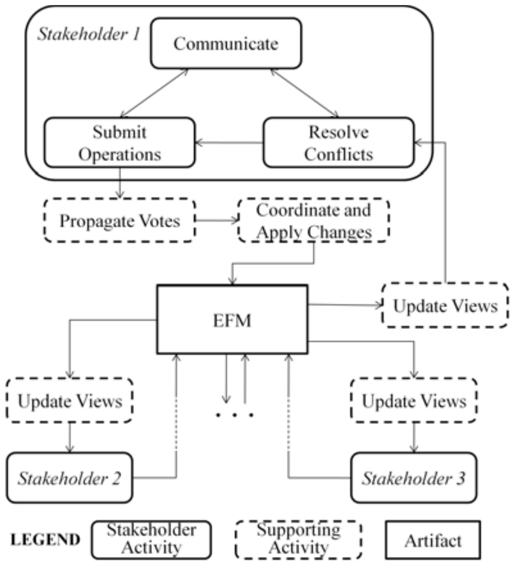


Figure 3. The process of collaborative feature modeling.

There are two categories of activities in it,

- Stakeholder Activity: activities initiated by stakeholders,
- Supporting Activity: activities automatically initiated by the system.

Activities of each category are described in Table II.

TABLE II. ACTIVITIES

Activity	Description
Stakeholder Activities	
Submit Operations	Submit creating and voting operations.
Resolve Conflict	Resolve conflict in the working view.
Communicate	Communicate with other stakeholders via comments and/or discussion pages.
Supporting Activities	
Propagate Votes	Propagate votes after operations are submitted.
Coordinate and Apply Changes	Coordinate changes made by stakeholders, and then apply them to the EFM.
Update Views	Update global, working and personal views.

According to the process, a stakeholder goes through the following steps iteratively.

Update Views: When a stakeholder starts to work, views are automatically generated for him. Whenever any stakeholder successfully updates the shared EFM, all stakeholders' views will be updated. Particularly, every update of the working view includes a detection of conflicts.

Resolve Conflicts: When the views are updated, stakeholders should first focus on the conflicts detected in their working views and resolve them using the creating and/or voting operations, and they need to communicate with others during the resolution. (See Section B.)

Submitting Operations: In addition to conflict resolution, stakeholders submit operations to construct the EFM, and communicate with others during the construction.

Propagate Votes: Propagated votes are computed after an operation has submitted. Then the original operation and propagated votes are ready to update the shared EFM.

Coordinate and Apply Changes: In the context of collaborative work, changes submitted by multiple stakeholders need to be coordinated. After the coordination, if the original changes are still valid, they are applied to the EFM and then cause the update of views of all stakeholders, otherwise the changes are neglected and its submitter will be informed. (See Section C.)

B. *Conflict Resolution*

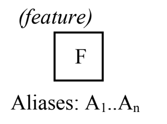
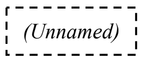
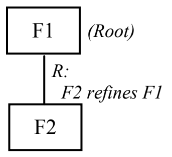
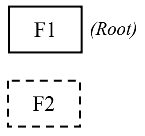
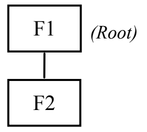
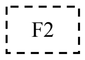
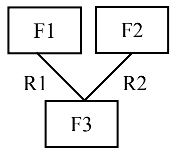
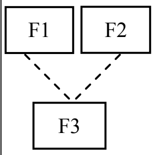
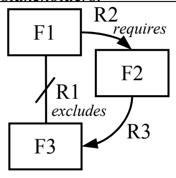
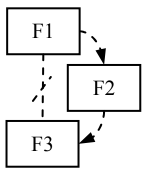
It has been demonstrated that conflict is common in collaborative work, because of divergence between stakeholders. However, not all divergence leads to conflicts, only significant ones do [7]. Particularly, in our divergence-tolerant approach, only the divergence in the working-view-level, not the EFM-level, can lead to conflicts. That is to say, we don't force stakeholders to eliminate any divergence in an EFM. In contrast, a stakeholder's working view is his individual workspace, and his work is considered unfinished if there is any conflict in it.

There are four types of conflict in one's working view.

- Unidentifiable Feature (UF): the stakeholder denies all existing names of a feature without giving a new one, which makes the feature unidentifiable in his working view. (Names are identifiers of features.)
- Non-positioned Feature (NPF): the stakeholder denies all existing positions of a feature without giving 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 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.

Since working views are generated through stakeholders' operations, these conflicts emerge with their operations as well. Table III gives examples of conflicts in a stakeholder's working view.

TABLE III. EXAMPLES OF CONFLICTS IN A WORKING VIEW

Type	The Stakeholder's		
	Initial WV	Operations	Resulting WV
UF		Vote no on the name "F"; Vote no on the name "A_1"; … Vote no on the name "A_n".	
NPF (1)		Vote no on R	
NPF (2)		Vote no on F1	
CR	 *R1, R2 might be created by different stakeholders.*	(None)	
CC	 *R1, R2, R3 might be created by different stakeholders.*	(None)	

The example shows that conflicts might emerge from operations of one (UF, NPF) or multiple (CR, CC) stakeholders. To resolve the conflicts, stakeholders also rely on the creating and voting operations.

For the unidentifiable feature, the stakeholder should either create a suitable name, or reconsider existing names.

For the non-positioned feature, the stakeholder should create a refinement involving it as a child, or make it a root feature explicitly, or reconsider existing refinements.

For conflicting refinements, the stakeholder should vote on these refinements to accept exactly one of them, or even denies all and create a new one.

For conflicting constraints, the stakeholder should vote no on some of them until the conflict has resolved, or even denies all.

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

C. *Coordination*

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, and their changes of the elements need to be coordinated. The need of coordination is rooted in the *update delay problem*: update of the EFM originated by one stakeholder cannot immediately become visible to others. According to types of the update (create or vote), there are four possible situations, as illustrated in Fig. 4.

Figure 4. Examples of situations needing coordination.

Duplicate Creation happens when a stakeholder (S2) creates an element E before a previous creation of the same element has been updated into the EFM and becomes visible to S2. Therefore the creation of the same element happens twice.

Conflicting Aliases emerge when two stakeholders try to create the same alias on different features at the same time. Since different features cannot have the same name or alias, the situation needs to be coordinated.

Unreachable Vote is the vote on an inexistent element. In Fig. 4, if the No vote on E submitted by S1 leads to the deletion of E, then S2's Yes vote is an unreachable vote.

Unreachable Propagation is similar. In Fig. 4, if the No vote on F1 leads to the deletion of F1, and if S2 creates a constraint involving F1 before the deletion becomes visible to him, then the propagation of Yes vote on F1 (according to rule PR-1) is unreachable.

Coordination of these situations follows a *serialized update strategy*, that is, all update applies to the EFM in the same order of their submitting. Furthermore, if the being applied update is no longer valid, it fails and takes no effect on the EFM, and its submitter will be informed the failure.

For duplicate creations, the first creation adds a new element to the EFM, and the later creations are converted to yes votes. In Fig. 5, S1 will create the element E, and S2 will have a yes vote on E.

For conflicting aliases, the first alias is kept, and the later ones are neglected. In Fig. 5, only Feature 1 will be assigned an alias N.

For unreachable votes, they are no longer valid on an inexistent element and are neglected. In Fig. 5, the element E will be deleted, and S2 will be informed his vote has failed.

For unreachable propagations, they are neglected, as well as the operations which cause the propagations. In Fig. 5, the feature F1 will be deleted, and S2 will be informed his creation of the constraint has failed.

D. *Guidelines for Stakeholders*

Perform conflict resolution as a starting point. When a stakeholder begins to work, the first thing he should do is to resolve conflicts, if any, in his working view.

Focus on divergence. Stakeholders should always pay attention to divergence, even if the divergence doesn't lead to conflicts, especially those they have not voted on. Divergence often indicates uncommon knowledge of a domain. By discussing and expressing opinions on them, stakeholders can improve their understanding of the domain.

Show support explicitly by voting yes. Although the vote propagation rule PR-CV ensures an initial Yes vote on elements from their creators, we recommend other stakeholders explicitly vote yes when they agree the existence of the elements. Thus the creators can tell that their contributions have been reviewed and accepted by others.

Use personal views to track different perspectives of a domain. Stakeholders can switch to anyone's personal view to get a snapshot of a specific perspective of the domain. For example, users may concern more about service-level features, while programmers might be interested in the implementation-level features and constraints between them. Thus it is possible that the personal views of users and programmers depict different perspectives of a domain.

Use global views to avoid information missing. If a stakeholder has voted no on an element, the element will be invisible in his working view since then. Therefore, he might miss useful information about the element without using the global view. For example, a stakeholder may vote no on a feature in early stage of the work. Since then, most stakeholders have expressed opposite opinions on the feature. This possibly means the first stakeholder needs to reconsider his decision. By using global views frequently, stakeholders would not miss information like this.

V. A Case Study

In this section, we introduce a case study to demonstrate the feasibility of our approach. In the study, an EFM of the video playing software domain is collaboratively constructed by stakeholders from different backgrounds. For simplicity, only three stakeholders are involved. Two of them are frequent users (U1 and U2), and the other one (Programmer P) has experience in developing video playing software.

U1 is the first contributor, as shown in Table III.

TABLE IV. The EFM at the Beginning (Part)

Elements in the EFM	Votes of Stakeholders

	U1
Features	
Play Control (F1), Play/Pause (F2), Stop (F3)	$\bigcirc_C$
Relationships	
(F2, F3) refines F1	$\bigcirc_C$

$\bigcirc_C$: YES votes initiated when creating (PR-CV).

After that, another user U2 joins and makes some changes on the EFM, as Table IV shows.

TABLE V. THE EFM AFTER U2 JOINS (PART)

Elements in the EFM	**Votes of Stakeholders**	
	U1	***U2***
Features		
Play Control (F1), Play/Pause (F2), Stop (F3)	$\bigcirc_C$	$\bigcirc$
Basic Control (F4), Advanced Control (F5), Repeat (F6), Jump (F7); Streaming Media (F8), Online Video Clip (F9), Online TV (F10)		$\bigcirc_C$
Relationships		
(F2, F3) refines F1	$\bigcirc_C$	X
(F2, F3) refines F4; (F6, F7) refines F5; (F4, F5) refines F1; (F9, F10) refines F8		$\bigcirc_C$

$\bigcirc$: YES votes submitted by voting operations.

X: NO votes submitted by voting operations.

Now, the working view of U1 highlights some worth noticing elements for him:

- Abnormal Features: F2 and F3 are multi-positioned because U2 created new refinements which U1 has not voted on.
- Controversial Refinements: "(F2, F3) refines F1" is supported by U1 but opposed by U2. (When U2 is creating "(F2, F3) refines F4", he also votes no on "(F2, F3) refines F1" otherwise F2 and F3 will be multi-positioned in his working view.)

After reconsidering the refinements, U1 solves these problems and achieves consensus with U2, by voting no on "(F2, F3) refines F1" and yes on other refinements. In addition, the "Advanced Control" created by U2 inspire U1 to add more features relating to it. However, U1 opposes the existence of "Online TV" because his experience tells him that online TV playing itself is a different domain. Changes made by U1 are shown in Table V.

TABLE VI. THE EFM UPDATED BY U1 (PART)

Elements in the EFM	**Votes of Stakeholders**	
	U1	***U2***
Features		
Play Control (F1), Play/Pause (F2), Stop (F3)	$\bigcirc_C$	$\bigcirc$
Basic Control (F4), Advanced Control (F5), Repeat (F6), Jump (F7); Streaming Media (F8), Online Video Clip (F9)	$\bigcirc_P$	$\bigcirc_C$
Online TV (F10)	X	$\bigcirc_C$
Slow (F11), Fast (F12)	$\bigcirc_C$	
Relationships		
~~(F2, F3) refines F1~~	X	X
(F2, F3) refines F4; (F6, F7) refines F5; (F4, F5) refines F1	$\bigcirc$	$\bigcirc_C$
F10 refines F8	X_p	$\bigcirc_C$
F9 refines F8	$\bigcirc$	$\bigcirc_C$
(F11, F12) refines F5	$\bigcirc_C$	

$\bigcirc_P$: YES votes propagated by PR-1 to PR-6.

X_p: NO votes propagated by PR-1 to PR-6.

Although it is just a very small part of the EFM, it still reflects important characteristics of our approach:

- Divergence Tolerance: Different viewpoints on the domain can be tolerated and coexist, and we can treat them as indicators of variability. For example, U1 and U2 have different opinions on "Online TV", which results in different personal views of the domain. Furthermore, it indicates that "Online TV" might be an optional child of "Streaming Media".
- Consensus Facilitation: By highlighting the abnormalities and controversies in working views, stakeholders are guided to focus on these elements, which help speed up the achievement of consensus.

In addition, the quality of the feature model is improved when more stakeholders join the construction. This can be demonstrated through the construction of video format/codec related features, before and after the programmer P joins.

Before the programmer joins, U1 and U2 have created some features about the video file formats. However, they confused video format with codec due to lack of technical expertise, as Table VI shows. (A video format can store data of different codec, e.g. "MP4" is a video format which can hold data of the codec MPEG-4 ASP or MPEG-4 AVC.)

TABLE VII. FORMAT/CODEC BEFORE P JOINS (PART)

Elements in the EFM	**Votes of Stakeholders**	
	U1	***U2***
Features		
Video Format (F1), AVI (F2), MP4 (F3)	$\bigcirc_C$	$\bigcirc$
MKV (F4), WMV (F5)		$\bigcirc_C$
Names (Aliases)		
F3: MPEG-4 (*confuse MP4 with MPEG-4*)		$\bigcirc_C$
Relationships		
(F2, F3) refines F1	$\bigcirc_C$	
(F4, F5) refines F1		$\bigcirc_C$

After the programmer joins, he clears up the confusion of format and codec, and adds constraints between them, thus the quality of the EFM is improved. (See Table VII.)

TABLE VIII. FORMAT/CODEC AFTER P JOINS (PART)

Elements in the EFM	Votes of Stakeholders		
	U1	*U2*	*P*
Features			
Video Format (F1), AVI (F2), MP4 (F3)	$\bigcirc_C$	$\bigcirc$	$\bigcirc_P$
MKV (F4), WMV (F5)		$\bigcirc_C$	$\bigcirc_P$
Video Codec (F6), MPEG-4 AVC (F7), MPEG-4 ASP (F8)			$\bigcirc_C$
Names (Aliases)			
F3: MPEG-4		$\bigcirc_C$	X
F4: Matroska; F7: H.264			$\bigcirc_C$
Relationships			
(F2, F3) refines F1	$\bigcirc_C$		$\bigcirc$
F4 refines F1		$\bigcirc_C$	$\bigcirc$
F5 refines F1		$\bigcirc_C$	X
(F7, F8) refines F6			$\bigcirc_C$
F6 mutual-requires F1 *(formats and codec must work together to deliver video data)*			$\bigcirc_C$

During the construction, the users often focus on service-level features and contribute little to constraints. In contrast, the programmer is able to add more implementation-level features and explore the constraints between existing features. Therefore the EFM becomes more comprehensive and useful with the help of collaboration between stakeholders.

Fig. 8 shows the EFM, in which the constraints and features in different personal views are marked. We can see that the personal views depict different perspectives of the domain. For example, when talking about "Streaming Media", only the "Protocol" matters in programmer' point of view; there's no difference between online TV and video clip. In contrast, users care about the choice between TV programs and video clips.

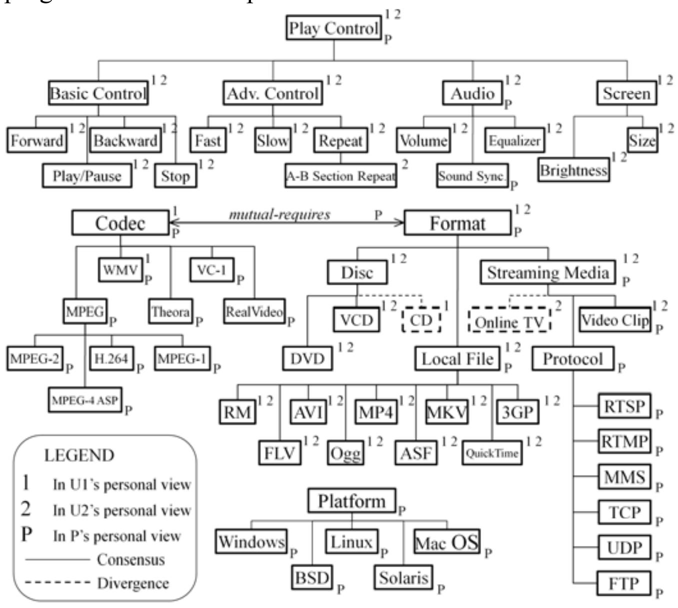


Figure 5. The EFM of video playing software domain.

VI. RELATED WORK

In FODA [14], FORM [15], FeatuRSEB [11], and FOPLE [16], feature models are constructed through a systematic analysis of commonality and variability in a family of systems across a domain. Such task is performed by domain analysts, with the assistance of other stakeholders such as users, domain experts and requirements analysts. However, despite its inherent need of collaboration, none of the methods has explicitly incorporated mechanisms or techniques for supporting collaboration in the construction of feature models.

On the other hand, there are several methods and tools for generic collaboration support. The IBIS (Issue Based Information System) method and the gIBIS tool [3] proposed a framework for people to solve complex problems in a structured way, that is, solve a problem by identifying Issues (what the problem is), submitting Positions (possible solutions to Issues), and providing Arguments (reasons to support or object to Positions), in a shared workspace. However, as a generic collaboration tool, there is no direct support for concepts of feature models, thus if domain analysts use gIBIS to gather information from other stakeholders, extra work must be done to translate an Issue-Position-Argument (IPA) model into a feature model. In fact, this is one problem that requirements analysts complain when they use gIBIS to elicit requirements [3]. If we map the conceptual framework of our approach into the IPA framework – constructing an EFM (the only Issue), creating elements (Positions), and voting on elements (Arguments) – the EFM is finished when stakeholders have settled the IPAs, thus no extra work needed.

In the research of requirements elicitation, researchers have already realized the importance of collaboration support. The WinWin approach [1] was proposed for collaborative requirements elicitation and negotiation involving multiple stakeholders. The Synoptic system [7] incorporated a similar process of negotiating multiple perspectives of requirements. The CREWS project [13][17] proposed a set of methods and tools for collaboratively eliciting and validating requirements with scenarios, although it focused much more on scenarios than on collaboration support. The OPCI method [2] incorporated data mining and recommender systems to improve forum-based collaborative requirements elicitation. In [4], the Wikipedia was introduced as a platform of stakeholders' collaboration during requirements elicitation, with enhancements like special document/page templates, page analysis, and wiki extension tools. Researchers in IBM [20] combined several tools, including forums, mailing lists, wikis and a specialized tool called TeamRoom, to help stakeholders express and discuss their requirements.

In addition, there exist several collaborative requirements management tools and methods. The EGRET tool [23] integrates communication support in traditional requirements management environment, enabling stakeholders to negotiate requirement changes without leaving their work environment. A web-based tool introduced in [17] allows stakeholders to upload local requirement documents or collaboratively write documents online, and then propose and discuss requirement changes through comments. References [19] and [4] have also discussed the importance of user participation and the need of collaboration tools in requirements management.

For other models, such as ontology, there exist some tools supporting collaborative model construction. The Ontolingua Server [10] provided a web server for collaborative ontology construction, and it uses an access control mechanism that is typical in most multi-user file systems: read and write access to ontology is controlled by its owner. Another example is OntoEdit [22], which forces users to lock before editing to avoid overwriting in a shared ontology. Like most collaboration tools, both tools emphasize the achievement of consensus and neglect divergence between users, or consider it harmful.

VII. Conclusions and Future Work

This paper proposes a collaborative feature modeling approach, in which stakeholders can create new elements in a feature model and vote yes or no on existing ones to express their opinions on the elements. Besides, we propose a collaborative feature modeling process, as well as a set of guidelines. A case study of video playing software domain shows the divergence tolerance and consensus facilitation characteristics of our approach, and also demonstrates that when more stakeholders join the collaboration, it is more likely to improve the quality of constructed feature models.

Our future work will focus on providing practical tool support for our approach, and apply it to a suitable environment, such as SaaS product development and maintain, where the number of stakeholders is larger and the collaboration is more intensive, to verify scalability and usability of our approach.

Acknowledgment

This work is supported by the National Basic Research Program of China (973) under Grant No. 2009CB320703, the Science Fund for Creative Research Groups of China under Grant No. 60821003.

References

[1] B. Boehm, P. Bose, E. Horowitz, M. J. Lee, “Software Requirements As Negotiated Win Conditions,” Proc. of Intl. Conf. Requirements Engineering (RE 94), 1994, pp. 74–83.

[2] C. Castro-Herrera, J. Cleland-Huang, B. Mobasher, “Enhancing stakeholder profiles to improve recommendations in online requirements elicitation,” Intl. Conf. on Requirements Engineering (RE 09), Sep. 2009, pp. 37–46.

[3] J. Conklin, M. Begeman, “gIBIS: A Hypertext Tool for Exploratory Policy Discussion,” ACM Trans. On Information Systems (TOIS), 1988, vol. 6, pp. 303–331.

[4] D. Damian, S. Marczak, I. Kwan, “Collaboration patterns and the impact of distance on awareness in requirements-centred social networks,” Intl. Conf. on Requirements Engineering (RE 07), Oct 2007, pp. 59–68.

[5] B. Decker, E. Ras, J. Rech, P. Jaubert, M. Rieth, “Wiki-Based Stakeholder Participation in Requirements Engineering,” IEEE Software, 2007, vol. 24, no. 2, pp. 28–35.

[6] P. Dourish, V. Bellotti, “Awareness and Coordination in Shared Workspaces,” Proc. of the 1992 ACM Conf. on CSCW, pp. 107–114.

[7] S. Easterbrook, “Handling Conflicts Between Domain Descriptions with Computer-Supported Negotiation,” Knowledge Acquisition: An International Journal, 1991, vol. 3, 255–289.

[8] C. A. Ellis, S. J. Gibbs, G. Rein, “Groupware: Some Issues and Experiences,” Communications of ACM, 1991, vol. 34, pp. 39–58.

[9] C. Ellis, J. Wainer, “A Conceptual Model of Groupware,” Proc. of the 1994 ACM Conf. on CSCW, pp. 79–88.

[10] A. Farquhar, R. Fikes, J. Rice, “The Ontolingua Server: a Tool for Collaborative Ontology Construction,” Intl. Journal of Human Computer Studies, 1997, vol. 46, no. 6, pp. 707–728.

[11] M. L. Griss, J.Favaro, M. d’Alessandro, “Integrating Feature Modeling with the RSEB,” Fifth Intl. Conf. on Software Reuse (ICSR 98), IEEE Computer Society, Jun. 1998, pp. 76–85.

[12] J. Grudin, “Computer-supported cooperative work: history and focus,” Computer, 1994, vol. 27, no. 5, pp. 19–26.

[13] P. Haumer, K. Pohl, K. Weidenhaupt, “Requirements Elicitation and Validation with Real World Scenes,” IEEE Trans. on Software Engineering, Dec. 1998, pp. 1036–1054.

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

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

[16] K. C. Kang, J. Lee, P. Donohoe, “Feature-Oriented Product Line Engineering,” IEEE Software, vol. 19, no. 4, July/Aug. 2002, pp. 58–65, doi:10.1109/MS.2002.1020288.

[17] M. Lang, J. Duggan, “A Tool to Support Collaborative Software Requirements Management,” Journal of Requirements Engineering, 2001, vol. 6, part 3, pp. 161–172.

[18] Y. Laurillau, L. Nigay, “Clover Architecture for Groupware,” Proc. of the 2002 ACM Conf. on CSCW, pp. 236–245.

[19] A. Rashid, “OpenProposal: towards collaborative end-user participation in requirements management by usage of visual requirement specifications,” Intl. Conf. on Requirements Engineering (RE 07), Oct. 2007, pp. 371–374.

[20] R. Recio, C. Salzberg, J. Palm, C. Machuca, “Leveraging collaborative technologies in the IO requirements process,” Intl. Conf. on Requirements Engineering (RE 08), Sep. 2008, pp. 283–288.

[21] C. Rolland, C. Souveyet, C. B. Achour, “Guiding Goal Modeling Using Scenarios,” IEEE Trans. on Software Engineering, Dec. 1998, pp. 1055–1071.

[22] K. Schmidt, C. Simone, “Coordination Mechanisms: Towards a Conceptual Foundation of CSCW System Design,”, CSCW Journal, 1996, vol. 5, no. 2–3, pp. 155–200.

[23] V. Shiha, B. Sengupta, S. Chandra, “Enabling Collaboration in Distributed Requirements Management,” IEEE Software, 2006, vol. 23, no. 5, pp. 52–61.

[24] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke, “OntoEdit: Collaborative Ontology Development for the Semantic Web,” Lecture Notes in Computer Science, 2002, pp. 221–235.

[25] W. Zhang, H. Mei, H. Zhao, “A feature-oriented approach to modeling requirements dependencies,” in Proc. of the 13th IEEE Intl. Conf. on Requirments Engineering (RE 05), 2005, pp. 273–282.