



Eliciting user requirements for e-collaboration systems: A proposal for a multi-perspective modelling approach

DOI:

[10.1007/s00766-017-0285-7](https://doi.org/10.1007/s00766-017-0285-7)
[10.1007/s00766-017-0285-7](https://doi.org/10.1007/s00766-017-0285-7)

Document Version

Accepted author manuscript

[Link to publication record in Manchester Research Explorer](#)

Citation for published version (APA):

Wang, Y., & Zhao, L. (2019). Eliciting user requirements for e-collaboration systems: A proposal for a multi-perspective modelling approach. *Requirements Engineering*, 24(0), 205–229. <https://doi.org/10.1007/s00766-017-0285-7>, <https://doi.org/10.1007/s00766-017-0285-7>

Published in:

Requirements Engineering

Citing this paper

Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights

Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy

If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [<http://man.ac.uk/04Y6Bo>] or contact uml.scholarlycommunications@manchester.ac.uk providing relevant details, so we can investigate your claim.



Eliciting user requirements for e-collaboration systems: A proposal for a multi-perspective modeling approach

Ye Wang¹ • Liping Zhao²

Abstract E-collaboration systems have become a new way of doing business that supports and enables communication, coordination and cooperation between people in shared projects and so on. Yet, eliciting user requirements for e-collaboration systems has proved to be a great challenge, due to the need to capture different knowledge for many different types of stakeholder. This paper proposes an approach for eliciting user requirements for e-collaboration systems. This approach takes a scenario description of e-collaboration systems as input and transforms it into a rich set of four-perspective requirement models, namely Coordination Model, Communication Model, Connection Model, and Collaboration Model. These models respectively represent the coordination, communication, collaboration, and connection characteristics of e-collaboration systems. Collectively, these models aim to provide a more complete and accurate requirements specification. An example from a collaborative office system is used to illustrate the applicability of the proposed approach. This paper also discusses practical implications of the proposed approach and then provides an assessment for the proposed approach.

Keywords E-Collaboration Systems, Model-Based Requirements Elicitation, Multi-Perspective Modeling, User Scenarios

Ye Wang
yewang@zjgsu.edu.cn
 tel: +86-18072711212

Liping Zhao
Liping.zhao@manchester.ac.uk

¹ Zhejiang Gongshang University, China

² The University of Manchester, UK

1 Introduction

In recent years, e-collaboration has become “a new way of doing business and a strategic weapon which could fundamentally change the traditional business relationships” [34][57]. E-collaboration is broadly defined as “virtual teaming of structured communication activities by using electronic tools e.g. blogs, groupware, discussion boards, portals and instant messaging” [15][48]. Based on the Internet, e-collaboration systems, under different synonyms such as groupware, group support systems, collaborative systems, cooperative information systems, have become an important instrument in supporting and enabling the communication, coordination and cooperation between people in shared projects, processes and teams within and across the organizations [30]. Some specific applications of e-collaboration systems, such as e-participation systems and e-procurements systems, have been used in a large number of domains [31][53], including education [3], transport [1], government [29], and healthcare [13].

With the rising economic importance of e-collaboration systems, there is a great demand on their development. Yet, there is still a lack of understanding of how best to develop these systems so that they can function according to the needs and requirements of their stakeholders [14][33]. According to Hickey and Davis [23], the technique used to elicit requirements greatly depends on the characteristics of the system-to-be. For e-collaboration systems, four basic characteristics – *Coordination*, *Communication*, *Collaboration* and *Connection* (also known as *4Cs*) – are identified [9][44]. Each of the characteristics addresses one critical aspect of an e-collaboration system [44]. This suggests that every phase in developing e-collaboration systems should support the *4Cs* [49]. Therefore requirements elicitation for e-collaboration systems should be based on these characteristics [19].

However, according to our research, very little work has been done to address all of the *4Cs* in the requirements elicitation for e-collaboration systems. For example, Bendjenna et al. [4] proposed MAMIE (from MAcro to MIcro level requirements Elicitation) to elicit requirements for inter-company cooperative information systems by coupling goals, scenarios and viewpoints. In MAMIE, goals are documented in terms of texts in each independent cooperation use case, which does not stress the connections between different stakeholders. Besides, the coordination

aspect of e-collaboration systems was not described in detail in MAMIE. The *i** framework is usually used to define requirements for systems in a collaborative environment [12][39]. However, it is generally understood that there is an absence of ordering information in the operationalization of the tasks in the *i** framework and a lack of the support for the coordination aspect of e-collaboration systems. Eliciting requirements for an e-collaboration system is difficult because it involves a large number of factors, such as the organization, context, problem, participants, processes and so on [49].

In order to support the elicitation of the requirements related to the *4Cs* of e-collaboration systems, we propose a multi-perspective modeling approach. By adopting a model-driven elicitation process, our approach aims to capture different aspects of e-collaboration systems and guide requirement engineers towards a more complete, accurate and reusable requirements specification. The proposed approach builds on our previous work on a pattern language for transforming scenarios into requirements models [59]. Scenario-based requirements engineering approaches [24][35] are widely used in requirements elicitation [68]. According to Lau [33], scenarios also play a crucial role in capturing user requirements for e-collaboration systems. The proposed approach takes an e-collaboration scenario description as input and transforms it into a set of requirement models, namely *Coordination Model*, *Communication Model*, *Connection Model* and *Collaboration Model*. While our approach currently only focuses on the functional requirements (*FRs*), the elicitation of non-functional requirements (*NFRs*) can be incorporated into the proposed approach using sound techniques such as Strategic Dependency Situations (*SDsituations*) [39] and the *i** framework [39][64]. The *SDsituations* uses constraints to capture the *NFRs* of a scenario and then transforms it into softgoals in the *i** framework.

The paper proceeds as follows. As a background to this paper, Section 2 describes the characteristics of e-collaboration systems. Section 3 provides the concepts and definitions of the proposed approach. Section 4 presents how the multi-perspective modeling approach can be coupled with e-collaboration scenarios. Section 5 demonstrates this approach through a real world example: a collaborative office system (*COS*). Section 6 evaluates our approach against some of the closely related ones by discussing the strengths and weaknesses of our approach.

Section 7 discusses how the proposed approach addresses the industrial challenges and its relevance to the industrial applications. Finally Section 8 concludes the paper.

2 Characteristics of e-collaboration systems

Although there is no single definition of “e-collaboration system”, such a system is generally characterized by four basic characteristics [9][44], also known as “4Cs”, which are:

- *Coordination.* An e-collaboration system offers people the ability to manage tasks, projects, workflows and appointments.
- *Communication.* An e-collaboration system enables people to communicate and exchange information via both asynchronous and synchronous modes.
- *Collaboration.* An e-collaboration system encourages people to work with each other for problem solving, with shared commitment and goals via knowledge sharing systems.
- *Connection.* An e-collaboration system enables people to connect with each other. The connection characteristic emphasizes the networking nature of the e-collaboration system.

The emphasis of the 4Cs suggests that an e-collaboration system and its environment should be defined to support 4Cs. Therefore, in developing an e-collaboration system, the requirements related to the 4Cs should be explicitly elicited and captured in the early phase of requirements engineering.

First, in order to support the coordination characteristic, requirements engineers need to elicit requirements for different knowledge of collaborative activities, such as the management of participants, their activities, workflows and so on.

Second, the communication characteristic requires requirements engineers to elicit the exchange of different types of messages between the system and its users participating in the collaboration process. Besides, a significant difference between e-collaboration systems and other types of systems is the number of users participating in an instance of a business process. In other types of systems, although there are multiple instances of users participating in a process, there is only one user performing the action in the system in a particular process instance. But in an e-collaboration system, the situation is different [7].

This difference is illustrated in Fig. 1 and Fig. 2. Fig. 1 illustrates an e-commerce process. In purchasing goods, there are two types of users: the customer and the seller. There can be multiple customers buying goods in the e-commerce system at the same time. But each user actually only participates in only one process instance of purchasing goods. A process instance that one user is involved will not affect the process instance of another user. They may face the problem of shared data but the workflows are independent. For e-collaboration systems, in a process instance, even for one role, there are usually multiple users participating in an activity at the same time. Fig. 2 illustrates an online video meeting system. In the process of create video meeting, one meeting participant

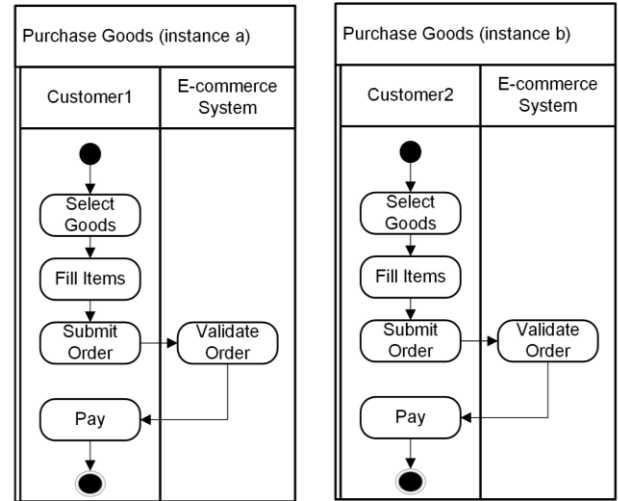


Fig. 1 Two independent instances of the Purchase Goods process participated by two customers

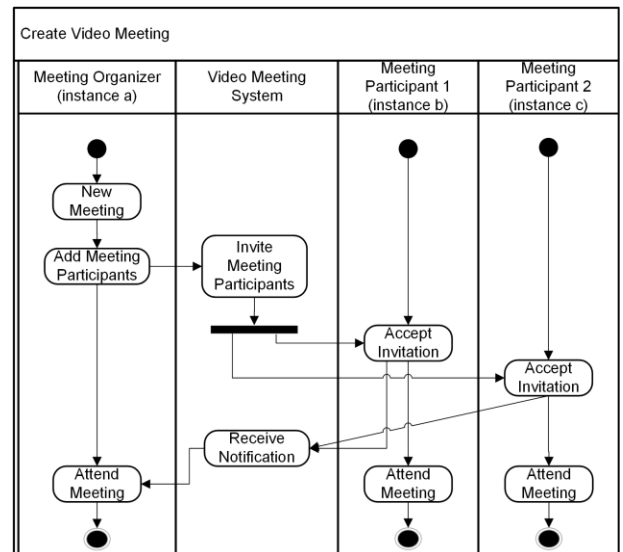


Fig. 2 The intersected process of Create Video Meeting with multiple instances involved

has been added to the meeting, a communication link will be established between him/her and other participants at once. The process instance of one user is intersected with, or even depends on the process instances of other users, which presents another challenge to requirements elicitation of e-collaboration systems.

Third, the connection characteristic requires requirements engineers to elicit different types of connections between agents. In e-collaboration systems, agents may be connected in order to achieve the same goal. One agent may depend on another agent to complete a task, or to transfer data. These different types of connections should be fully captured by requirements engineers.

Last but not least, each participant in the e-collaborative system should have intentionality [39]. The collaboration characteristic requires requirements engineers to understand and elicit the common goals of the participants. Besides, the way to achieve a goal (i.e., the tasks they need to perform) should also be captured. Therefore, the relationships between different tasks should be elicited as well.

The above description shows that in order to elicit the complex requirements of e-collaboration systems, we need a more comprehensive and systematical approach. In the remaining paper, we propose such an approach.

3 Concepts and definitions

3.1 The four-perspective models

To address the challenges with requirements elicitation in e-collaboration systems, we propose an approach to capture and model the requirements from multiple perspectives. Under this approach, user requirements are elicited by means of the four-perspective models, each addressing a characteristic of 4Cs. Fig. 3 summarizes these models and

their supporting characteristics. In what follows, we define each of these four models.

1) Coordination Model

Coordination Model coordinates the dynamics of collaborative work through a set of control flows in a process-centric view and sketches the story line of team working. This perspective captures both the domain and the system knowledge at different levels of abstractions, including the activities involved by the system and its users, the workflows among users and the system as well as the internal workflows performed by each user or the system. Therefore this perspective addresses the coordination characteristic of e-collaboration systems.

Fig. 4 (a) depicts the metamodel of Coordination Model and the concepts it used. Coordination Model consists of five fundamental elements [41]: 1) *activity*, which refers to the dynamic process. The execution of an activity is modeled as activity nodes connected by activity edges; 2) *activity node*, which denotes the steps of an activity. It covers two types of nodes in our approach, i.e., *action* and *control node*; 3) *action*, which represents a single step within an activity; 4) *control node*, which is an activity node used to coordinate the flows between other nodes. The control node is divided into *initial node*, *final node*, *fork node*, *join node*, *decision node*, and *merge node*. An initial node denotes a starting point for executing an activity. A join node is to synchronize multiple workflows. A merge node is to bring together multiple alternate workflows. A decision node is to choose between outgoing workflows. A fork node is to split a workflow into multiple concurrent workflows. A final node denotes the final state of an activity; 5) *activity edge*, which organizes the actions performed in the description through time. The metamodel of Coordination Model is adapted from the metamodel of UML Activity Diagram [41] by tailoring some unnecessary elements and remaining the most important elements.

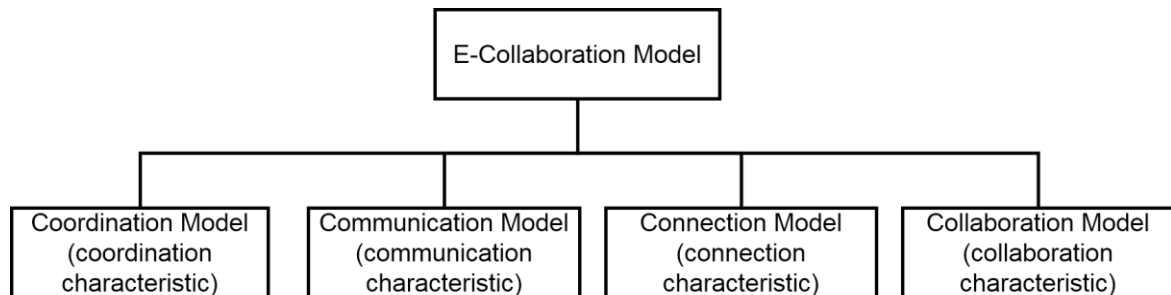
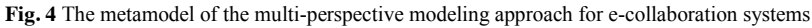


Fig. 3 The structure of the basic models of our approach



In an e-collaboration system, communications among users and the system are the most typical characteristic. Communication Model unearths human factors and the interactions that should be addressed in the e-collaboration work through a set of messages in an agent-centric view. Communication Model is used to represent the users and the system involved in the e-collaboration process and their interaction messages. Therefore this perspective addresses the communication characteristic of e-collaboration systems. The sequence and concurrency of each interaction between users and the system are also captured by Communication Model.

properties: 1) *action*, the action to send or receive a message, 2) *parameter*, the objects carried by a message, 3) *sequence number*, the sequence number of the message in an interaction, 4) *condition*, under which condition the message can be executed, 5) *concurrency*, denotes whether the message is executed sequentially or concurrently, 6) *message sort*, the type of message. The messages can be divided into three types:

- If the type of a message is not defined, the message is

only a normal send message. The metamodel of Communication Model is adapted from the metamodel of UML Communication Diagram [41] by keeping the most crucial elements such as the message and its properties such as the action, parameter, sequence number, message sort, and condition. In addition, we add the collaboration-specific property such as concurrency to the original metamodel.

3) Connection Model

Connection Model helps to discover different types of dependencies between users and the system through a set of links. Therefore this perspective addresses the connection characteristic of e-collaboration systems.

As shown in Fig. 4(c), Connection Model consists of *agents*, their *dependencies* and *dependums*. Dependency is a dependent relationship between two agents based on a dependum. In Connection Model, each node represents an agent, and each dependency between two agents indicates that one agent (i.e., the *dependor*) depends on the other (i.e., the *dependee*) for something (i.e., the *dependum*) in order that the former may attain some goal. There are three basic types of dependum in Connection Model: *task*, *goal* or *resource*. According to the dependum type, the dependencies are different. The task dependency means that one agent depends on another agent to complete a task. The goal dependency means that one agent depends on another agent to achieve a particular goal. The resource dependency means that one agent depends on another agent to provide a specific resource. The metamodel of Connection Model is adapted from the i* metamodel [16] and Connection Model can be represented by the *Strategic Dependency (SD) model* of the i* framework [65]. The SD model focuses on describing the network of relationships among agents and the agents' external intentional relationships with each other through the resource dependency, task dependency or goal dependency, which is quite fit for describing the connection characteristic of e-collaboration systems.

4) Collaboration Model

The intentionality of each user or system is crucial for e-collaboration systems because the collaboration made by two agents is driven by some common goals. Therefore it is necessary for e-collaboration systems to address the intentions or goals of each user and the system. Collaboration Model discovers the implicit intentionality of each user and system, and the shared problems that they have as well as the way how they solve the shared problem. This perspective captures the goals of each user and system,

and the tasks they performed to achieve each goal. Therefore this perspective addresses the collaboration characteristic of e-collaboration systems. Collaboration Model is a combination of the goal-centric model and the task-decomposition model.

As shown in Fig. 4(d), Collaboration Model consists of several *internal elements* and their *links*. The internal elements belong to different *agents* and thereafter outline the boundary of each agent. A *goal* unravels each agent's intentions, which is the core concept of Collaboration Model. Each agent achieves their goals by performing a set of *tasks* and making uses of *resources*. The *internal elements* are linked by two kinds of *links*: the *means-end* and *decomposition*. Decomposition represents that a task can be decomposed into a set of subtasks or subgoals, whereas means-end represents that there are different possible ways to accomplish a goal. The internal elements in one agent may be connected to internal elements in another agent via the *dependency*. The metamodel of Collaboration Model is also borrowed from the i* metamodel [16] [64].

To support the collaboration characteristic of e-collaboration systems, the *Strategic Rationale (SR) model* of the i* framework [65] can be used. The SR model describes and supports the reasoning that each agent has about its relationships with other agents. The SR model can be treated as a refinement of the SD model by elaborating the inherent intentions and the reasoning of the tasks of each agent. Therefore the SR model is well suited for addressing the collaboration characteristic.

In the SR model, each agent in the SD model is decomposed to a set of nodes, which may be goals, tasks, or resources. Through the creation of the SR model, one can gain insight into the goals of each agent and the means to implement the goals through the means-end link.

Table 1 summarizes the modeling elements of these four models.

3.2 Relationships between the models

The four-perspective models can be composed into one coherent model to represent the system as a whole. Fig. 5 shows the four types of relationship that connect the models. These relationships are described as follows.

(1) *contains*: the interactive actions in Coordination Model are treated as the property of messages in Communication Model. Compared to actions, messages

have more properties, such as the parameter, condition, and the concurrency constraint. We merge some repeated actions for each agent in Coordination Model but keep

them intact in Communication Model. The reason is that messages carry data but actions not. By carrying different data, the messages differ from each other. For example,

Table 1 The modeling elements of the four-perspective models

ID	Name	Definition	Related Model
1	Activity	An activity is a dynamic process connected by activity edges.	Coordination Model
2	Activity Edge	An activity edge represents the control flow from one activity node to another.	Coordination Model
3	Activity Node	An activity node denotes a step of an activity.	Coordination Model
4	Action	An action represents a single step within an activity.	Coordination Model
5	Control Node	A control node is an activity node used to coordinate the workflows between other nodes.	Coordination Model
6	Initial Node	An initial node denotes a starting point for executing an activity.	Coordination Model
7	Final Node	A final node denotes an end point to stop a workflow in an activity.	Coordination Model
8	Decision Node	A decision node is a node to choose between outgoing workflows.	Coordination Model
9	Join Node	A join node is a mediate node to synchronize multiple workflows.	Coordination Model
10	Merge Node	A merge node is a mediate node to bring together multiple alternate workflows.	Coordination Model
11	Fork Node	A fork node is a mediate node to split a workflow into multiple concurrent workflows.	Coordination Model
12	Agent	An agent is a user or a system that communicates with each other through exchanging messages. Agents are connected by dependencies and collaborating with each other.	Communication Model Connection Model Collaboration Model
13	Message	A message is a named element that defines one specific kind of communication between lifelines of an interaction.	Communication Model
14	Message Sort	Message sort is the type of a message.	Communication Model
15	Dependum	A dependum is a fact for which one agent depends on another.	Connection Model Collaboration Model
16	Goal	A goal is the intention of one or more agents. A goal is also a type of dependum, which means that one agent depends on another agent to achieve a particular goal.	Connection Model Collaboration Model
17	Task	A task is an activity that needs to be carried out. It is a type of dependum, which means that one agent depends on another agent to complete a task.	Connection Model Collaboration Model
18	Resource	A resource is the provision of some entity, physical or informational. It is a type of dependum for which one agent depends on another agent to provide a specific resource.	Connection Model Collaboration Model
19	Dependency	Dependency is a dependent relationship between two agents based on a dependum.	Connection Model Collaboration Model
20	Internal Element	An internal element is an abstract element which can be a goal, task or resource. A set of internal elements outline the boundary of each agent.	Collaboration Model
21	Link	A link is a fact by which two internal elements connected.	Collaboration Model
22	Decomposition	Decomposition represents that a task can be decomposed into a set of subtasks or subgoals.	Collaboration Model
23	Means-End	Means-end represents that there are different possible ways to accomplish a goal.	Collaboration Model

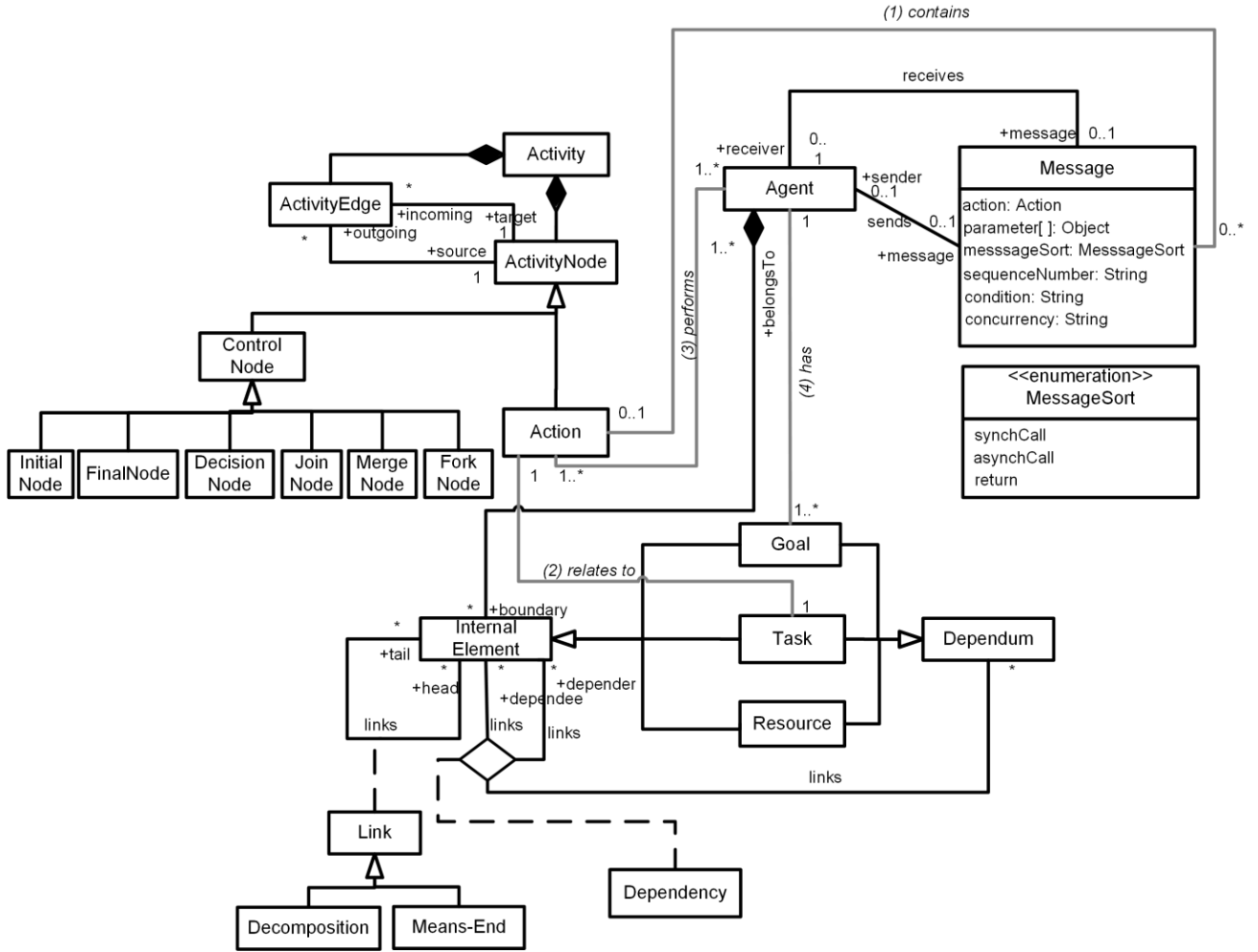


Fig. 5 The relationships between the four-perspective models

message – “scheduleMeeting (MeetingName)” can be treated as the same action “schedule meeting” with different meeting name.

(2) *relates to*: the atomic actions in Coordination Model are related to tasks in Collaboration Model at the same level, and vice versa. Note that composite actions are related to tasks in Collaboration Model at the level below. This suggests a close relationship between Coordination Model and Collaboration Model: Coordination Model controls the tasks specified in Collaboration Model. Yu also discussed this inter-relationship in [63].

(3) *performs*: denotes that each agent performs actions.

(4) *has*: denotes that each agent has one or more goals in the e-collaboration processes.

In addition, agents in Communication Model are mapped to agents in Connection Model at the same level. The difference between Communication Model and Connection Model is that Communication Model addresses the dynamic interactions between agents whereas Connection Model addresses the structural dependencies between agents.

4 The proposed approach

4.1 E-collaboration scenarios

In Requirements Engineering (RE), scenarios have been widely used as a communication tool to help software stakeholders and requirements engineers to describe and elicit requirements [21][26]. The narrative of the scenario makes it natural for people to express their needs and wants. Scenarios can also be used to capture the domain knowledge from the users and to communicate that knowledge to different software stakeholders. The scenarios technique has also shown its effectiveness in capturing requirements for e-collaboration systems [33].

According to Rolland et al. [45], a system’s scenarios can be divided into three levels: the *context level*, the *system interaction level* and the *system internal level*. The scenarios at a high-level can be refined into low-level scenarios [46]. The context level normally contains one scenario (i.e. *contextual scenario*) and this high-level

scenario should describe the owner's vision about the system to be developed. The system interaction level contains one or more scenarios (i.e. *system interaction scenarios*), each of which corresponds to a subgoal discovered at the contextual level. Scenarios at this level describe the interactions between the system and its users. These scenarios refine the contextual scenario and hence are more concrete. The system internal level contains several scenarios, each of which corresponds to a subgoal discovered at the system interaction level. Scenarios at this level describe the detailed system operations that support the interactions at the level above. These scenarios refine the system interaction scenarios and terminate the refinement relationship. The progress from one level to the next is determined by the goal decomposition in the goal model. Each goal leads to a new scenario description at a level below. We therefore said that it is a model-driven process based on a system of scenarios.

Inspired by Rolland et al. [46], we have adapted these three levels of scenarios to our approach and used them to guide requirements engineers to elicit requirements for e-collaboration systems. According to Yu [65], capturing requirements in a collaborative environment should put more emphasis on the intentionality of all agents, rather than the system. Therefore in our approach, we change Rolland et al.'s three levels of scenarios into: the context

level, agent interaction level and agent internal level. An e-collaboration scenario can be a *contextual scenario*, an *agent interaction scenario* or an *agent internal scenario*, each of which addresses different aspects of requirements in e-collaboration systems. The contextual scenario outlines the services that an e-collaboration system provides to its users, rather than how it provides the services. The agent interaction scenario describes how each e-collaboration service is provided by different agents in more detailed ways. The agent internal scenario describes how each agent works internally, including both the users and the system. Through the three types of scenarios, our approach allows requirements engineers to codify the interaction of human and computer agents at different levels of abstraction.

The metamodel of e-collaboration scenarios is shown in Fig. 6, which is adapted from [39][59]. An e-collaboration scenario is identified by one name (*title*) and has a *description*, which consists of an initial state (*source*), a flow of actions (*path*) and a final state (*goal*). The initial state defines the precondition for a scenario to be triggered and the final state defines the goal state reached at the end of the scenario. A goal state is defined as something that some stakeholder hopes to achieve in the future [43]. An e-collaboration scenario is participated by one or more *agents*, which can be actors involved in the scenario or the system. Agents perform some *action* and use some

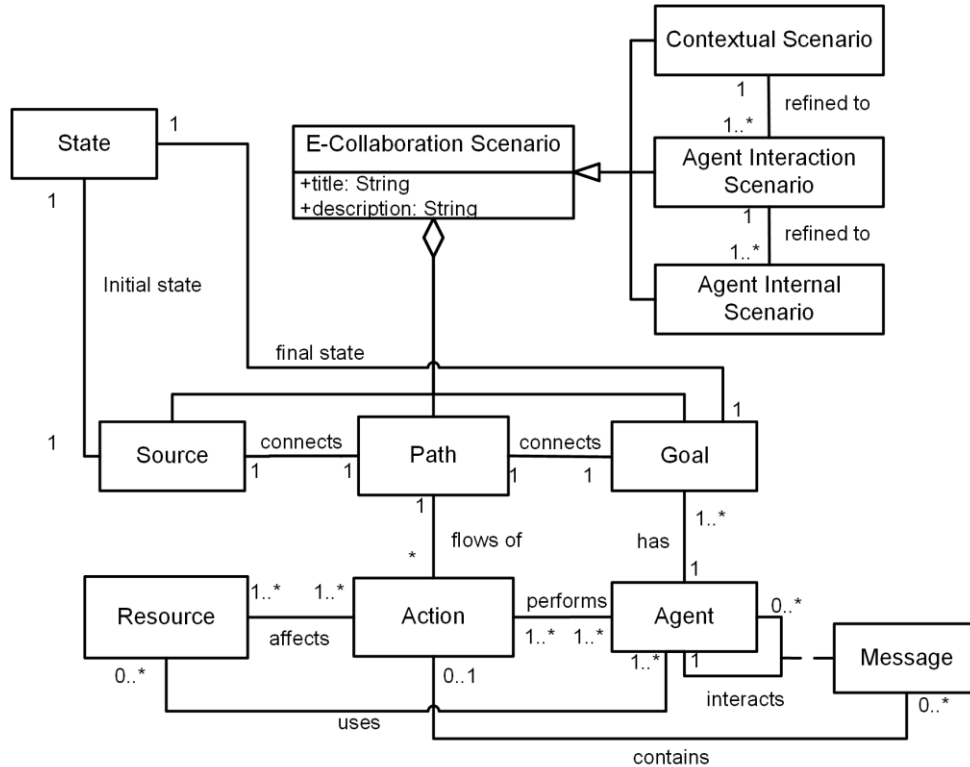


Fig. 6 The metamodel for e-collaboration scenarios

resources on the action individually or interactively to achieve the goal state. Each action may affect some resource. Agents interact with each other via *messages*, which may contain an action. It is interesting to find that the relationships between the four models (Line 1, 3 and 4 in Fig. 5) can be mapped to the links in the structure of e-collaboration scenarios (Fig. 6).

4.2 Defining and representing e-collaboration scenarios using the four-perspective models

Based on the approach proposed by Rolland et al. [46], we combine the four-perspective models with e-collaboration scenarios into a multi-perspective approach to requirements elicitation.

In our approach, agent interaction scenarios are derived from the connection model at the context level and authored by requirements engineers that follow a set of instructions and heuristics. Goals in the connection model at the context level correspond to agent interaction scenarios at the middle level, which capture the interactions between the e-collaboration system and its users. Agent internal scenarios are derived from collaboration models at the level above and authored by requirements engineers that follow a set of instructions and heuristics. Goals in collaboration models at the agent interaction level correspond to agent internal scenarios at the bottom level. In doing so, both goals and scenarios are refined across three levels. In addition to the goal and scenario refinement, the dependencies between agents and composite actions can also be refined across levels. A dependency in a connection model or a collaboration model can be refined to more concrete dependencies at the level below, whereas a composite action in a coordination model can be refined to atomic actions at the level below.

Since scenarios at three levels address different aspects of requirements knowledge, it is suggested that requirements engineers create different models for different levels:

1) *Create Connection Model at the context level.* Connection Model helps with capturing the high-level structural dependencies between agents described in the contextual scenario.

2) *Create the four-perspective models at the agent interaction level.* At this level, the scenario description needs to explicitly describe the interactions between

different agents. To fully understand the interactions, the dynamics of interactions are crucial, such as the workflows and the exchanging messages. Therefore, Coordination Model and Communication Model are created. The workflows are transformed to elements in Coordination Model, whereas the messages are transformed to elements in Communication Model. Besides, to assist requirements engineers to derive agent internal scenarios at the next level based on goals, it is necessary to analyze the purposes of interactions and agents. Thereafter, Connection Model and Collaboration Model are created. The intentionality of each agent involved in the interaction is transformed to elements in Connection Model and Collaboration Model

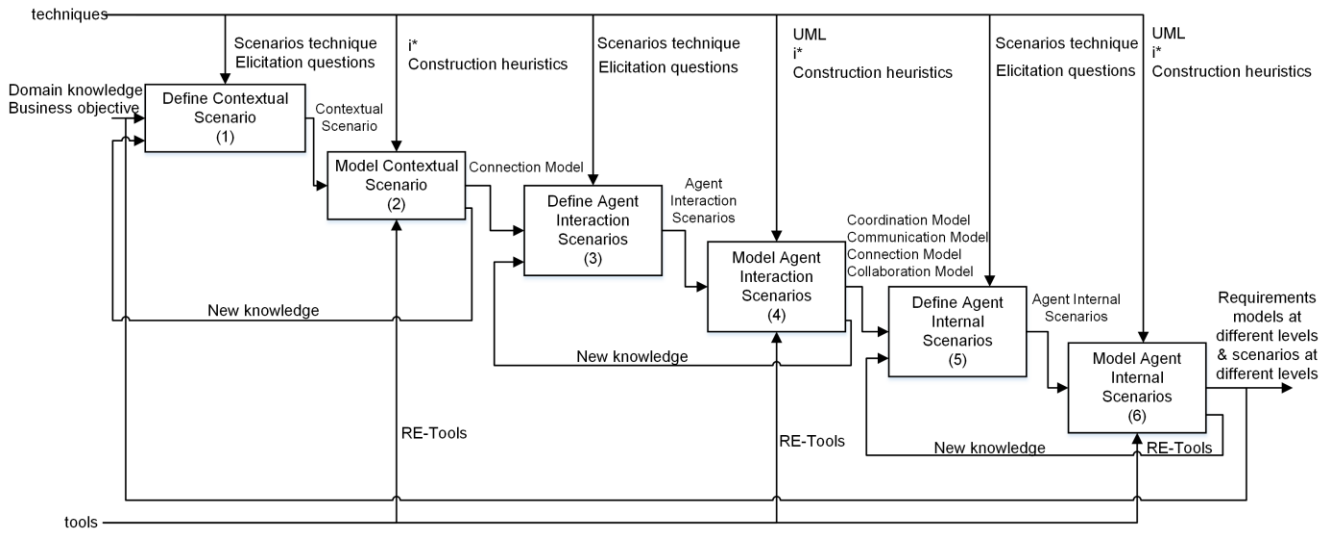
3) *Create Coordination Model and Collaboration Model at the agent internal level.* Scenarios at this level describe how each agent works internally, therefore the implicit goals and internal operations of each agents are important for this level. Coordination Model and Collaboration Model are created to capture agent internal requirements at this level. The implicit goals and internal operations are transformed into internal elements in Collaboration Model. Moreover, the dynamics between the internal operations are transformed into the elements in Coordination Model.

Table 2 shows the detailed steps of using the four-perspective models for different levels of scenarios and the corresponding outcome for each step. The elicitation process based on activities is also shown in the SADT diagram [47] in Fig. 7.

Step 1 is to define the contextual scenario. Based on this scenario, Connection Model is constructed (Step 2), which has two uses: 1) to derive agent interaction scenarios, 2) to provide a feedback on the contextual scenario and check whether there are new knowledge that need to be added to the scenario description. Step 3 is the definition of agent interaction scenarios and Step 4 is to model agent interaction scenarios by creating the four-perspective models. The four-perspective models can also be used to validate and improve the scenario descriptions. Coordination Model, Communication Model, Connection Model and Collaboration Model are created to capture the workflows, messages and the purpose of agent interactions. Step 5 is the definition of agent internal scenarios and Step 6 is to model agent internal scenarios by creating Coordination Model and Collaboration Model. Likewise, there is a feedback link from Step 6 to Step 5. Collaboration Model and Coordination Model are created to capture agent internal requirements, such as the implicit goals and

Table 2. The process of using the four-perspective modeling approach to requirements elicitation for e-collaboration systems

Step	Purpose	Outcome
Step 1: Define the contextual scenario	Elicit requirements at the	The contextual scenario
Step 2: Model the contextual scenario	context level	The connection model
Step 3: Define agent interaction scenarios	Elicit requirements at the agent	Agent interaction scenarios
Step 4: Model agent interaction scenarios	interaction level	Coordination models
Step 4.1: Create Coordination Model		Communication models
Step 4.2: Create Communication Model		Connection models
Step 4.3: Create Connection Model		Collaboration models
Step 4.4: Create Collaboration Model		
Step 5: Define agent internal scenarios	Elicit requirements at the agent	Agent internal scenarios
Step 6: Model agent internal scenarios	internal level	Coordination models
Step 6.1: Create Coordination Model		Collaboration models
Step 6.2: Create Collaboration Model		

**Fig. 7** The SADT diagram that describes the requirements elicitation process from four perspectives

internal operations of each agent.

For Step 1, 3 and 5, the scenarios technique and a set of questions can help to define and author the scenario at these levels. Note that we will design different questions for different levels. For Step 2, 4 and 6, RE-Tools [50] is employed as a modeling tool. Besides, the i* modeling framework can help to create Connection model and Collaboration model, whereas for Step 4 and 6, the UML modeling language is used to create Coordination model and Communication model. We are aware that UML and i* are not the only one technique to represent the four-perspective models. For example, BPMN can be used to represent Coordination Model [60]. Message Sequence Chart can be used to represent Communication Model [54]. CSRML (Collaborative Systems Requirements Modeling Language) [52] can be used to represent Connection Model and Collaboration Model. Requirements engineers can

choose the most familiar technique by borrowing its metamodel. It is worth noting that the elicitation process can be iterative. The outcome of Step 6 can be used for the system design, and moreover, as the input for the next-iteration requirements elicitation.

5 An example: applying the proposed approach to a collaboration office system

We have applied our approach to a real world e-collaboration application – a Collaborative Office System (COS) for requirements elicitation. COS is an enterprise service system supporting collaborative offices in China. The major users of the system are the members who have participated in a collaborative project. COS is a collaboration tool that facilitates the users to organize their projects into online boards. The following sections

demonstrate how the proposed approach is used to elicit a system of requirements from COS. These sections are based on the steps in Table 2.

5.1 Define the contextual scenario (Step 1)

To acquire the domain knowledge for the contextual scenario, the requirements engineer starts by finding out the main stakeholders involved in the domain and conducting an open-ended interview with all the stakeholders. During this meeting, the requirements engineer will try to get an initial idea of the domain by asking the stakeholders two questions:

- 1) Why do you need this system?
- 2) What services do you expect the system to provide for you?

By answering these two questions, the requirements engineer can construct the high-level domain knowledge for the system, and then write the contextual scenario. We do not require the contextual scenario to be written in a template, but it should contain at least two pieces of information: 1) the actors that use the system; 2) the services the system should provide to each actor. Other knowledge are welcome but not mandatory, such as the constraints for running the system, the resources used in the scenario, and so on. By following this guideline, the contextual scenario for COS contains the following statement:

“There are two types of users in the COS: the project owner and the project member. The COS should provide the following services to the project owner after he/she login the system: project creation, project management, notification management, and meeting scheduling; and provide the following services to project members: project management, notification management and meeting scheduling.”

Once the contextual scenario is defined, it can be validated by stakeholders to check whether there are additional knowledge to complement the scenario. The incorrectness and inconsistency can be found in this validation. In this step, a domain vocabulary is required to be first built among all stakeholders to facilitate their communication [12], but this is not the focus of this paper.

5.2 Model the contextual scenario (Step 2)

The next step for the requirements engineer is to model the

contextual scenario using Connection Model (i.e., the SD model in this paper). There are several heuristics for constructing the connection model:

1. Pick up actors and the system from the contextual scenario and map them to the agents in the SD model. In the i* model, there are different types of actors, such as roles, positions and agents [65]. Differentiating the actor types makes the definition more accurate but introduces too many concepts and makes it harder to model scenarios at different levels from different perspectives. Therefore in this paper, we only use agents as the core concept for e-collaboration scenarios. In the contextual scenario for COS, there are three agents: the project owner, project member and COS.
2. Analyze and model dependencies between agents. The services provided by the system are mapped to the goal dependency between the system and its users. For example, the COS provides the project management service to project members, then project members depend on the COS to achieve the “Project Be Managed” goal. To build the dependencies, the requirements engineer needs to ask each identified user the following questions:

- 1) Who do you need to depend on in doing your job?
- 2) Who need to depend on you to do their job?
- 3) In what way do you depend on other agents?

These questions are elaborated from [12]. These questions help to find out the basic dependencies between different types of agents. For example, the project owner relies on project members to participate in the project and to attend the meeting. Note that we adopt the goal formulations of the i* framework [64] in this paper. But there are a lot of other goal formulations [46][56] that can be used in Connection Model as well as Collaboration Model.

3. Create and validate the SD model. By using the RE-Tools [50], the requirements engineer produces the SD model (Fig. 8) according to the domain knowledge acquired from the above steps and then validates it with stakeholders to assure its correctness. Each goal (between the system and its users) at the context level suggests a new scenario. For example, in Fig. 8, the goal “Project Be Created” can only be realized in the system if more information about how

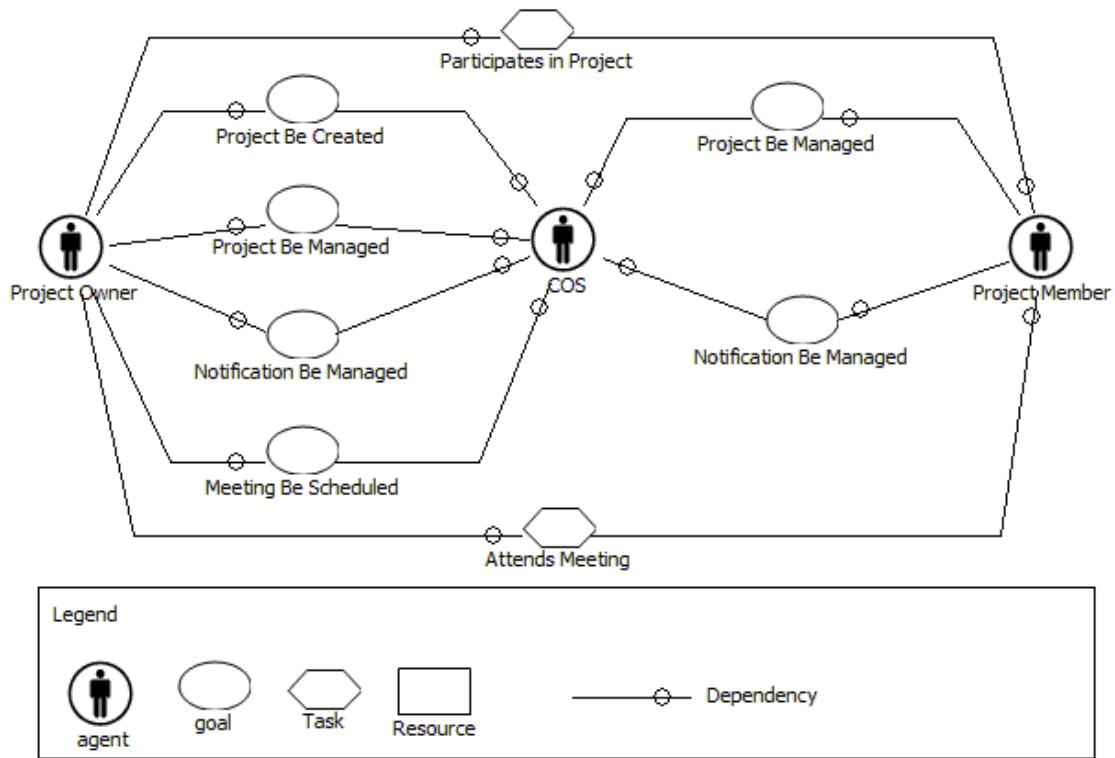


Fig. 8 The connection model at the context level

to achieve this goal is provided. At this point the requirements engineer is ready to move to the agent interaction level.

5.3 Define agent interaction scenarios (Step 3)

The agent interaction level contains several agent interaction scenarios, each of which corresponds to a goal discovered at the context level. In this paper we only illustrate the elicitation process for the “Project Be Created” goal at the context level.

To define agent interaction scenarios, the requirements engineer first needs to interview the stakeholders and acquire the domain knowledge from the stakeholders by asking the following questions for each goal:

- 1) To achieve this goal, what tasks do you need to perform, or what subgoals do you need to realize?
- 2) What are the relationships between these tasks or subgoals?
- 3) Do you need someone to provide some information before you do a task?
- 4) What response should the system give to you after you execute a task?
- 5) Can you perform this task at the same time with other people?

6) Are there any constraints or conditions for you to execute this task?

7) Who needs the outcome of your task?

By answering these questions, the requirements engineer can acquire the knowledge for the interactions between the system and its users, such as the individual and interactive actions performed in the scenario, the flows and conditions of these actions, system responses, resources, outcomes and concurrency constraints. Then the requirements engineer encodes these knowledge into the agent interaction scenario in a temporal manner. Below is the statement of one scenario which is named *Project Creation* corresponding to “Project Be Created” goal.

“After the user logs in the system, the user needs to create a new project by first entering a new project name. When the project is initially created, the system displays the new created project to the user. The user becomes the owner of the project. Next the system asks the owner to invite two project members to work on this project. It is required that all members have already registered. In order to add a member, the owner needs to obtain the personal information (i.e., the email or the phone number) of the member. Once a member receives an invitation from the owner, the project list of the member will be refreshed instantly with the new project updated. The owner next adds one stage named “planning”, and then adds another

stage “working on” to the project. The members can also add stages to the project. No matter who adds the stage to the project, the stage list of all members including that of the owner will be refreshed instantly with the stage added. The stages are initially ordered according to the creation time. After the stages are added, the owner creates one task at the planning stage and another task at another stage. Once the owner or the member adds a new task to the stage, the task list of all members will be updated instantly.”

Once an agent interaction scenario is defined, it can be validated by stakeholders, to check if the scenario contains any incompleteness, incorrectness and inconsistency.

5.4 Create Coordination Model (Step 4.1)

Coordination Model can be directly built according to the scenario description without asking the stakeholders more questions. There are several construction heuristics for constructing the coordination models:

1. Extract the source and goal of the scenario, and pick up actions of different agents. The source and goal can be mapped to the initial node and final node of the model respectively.
2. Organize all actions using activity edge and control nodes in a temporal manner (Fig. 9). Some tasks can be mapped to composite actions in Coordination Model, which will be refined into a set of actions at the level below. In the scenario example, there are multiple agents performing repeated actions. For example, two actions – “Add Stage” and “Create Task” – are performed more than one once in this scenario by different agents. To reduce the complexity

of each coordination model, we only consider the generic process without regard to the agents and thus merge repeated actions into one single action. In the Project Creation scenario, eight actions are identified below: “Enter Project Name”, “Display New Project”, “Invite Project Member”, “Update Project List”, “Add Stage”, “Update Stage List”, “Create Task”, and “Update Task List”. The requirements engineer structures the eight actions with the sequence flow in a temporal order in UML Activity Diagram [41]. The resulting coordination model is shown in Fig. 9.

5.5 Create Communication Model (Step 4.2)

Agents and messages are core to Communication Model at this level. In order to create a communication model, the requirements engineer needs to:

1. Identify agents from the scenario. In the example, the identified agents are the project owner, project member, and COS.
2. Identify messages from the scenario, including the properties of messages, such as the actions, parameters, message sort, and condition and concurrency constraints. Most messages can be extracted from the interactive tasks in the scenario, yet some properties may not be described in the scenario, which needs to be further clarified by the requirements engineer and the stakeholders by answering the following questions:
 - 1) Is this message a synchronous message, asynchronous message or return message?
 - 2) Is there any condition for you to send or

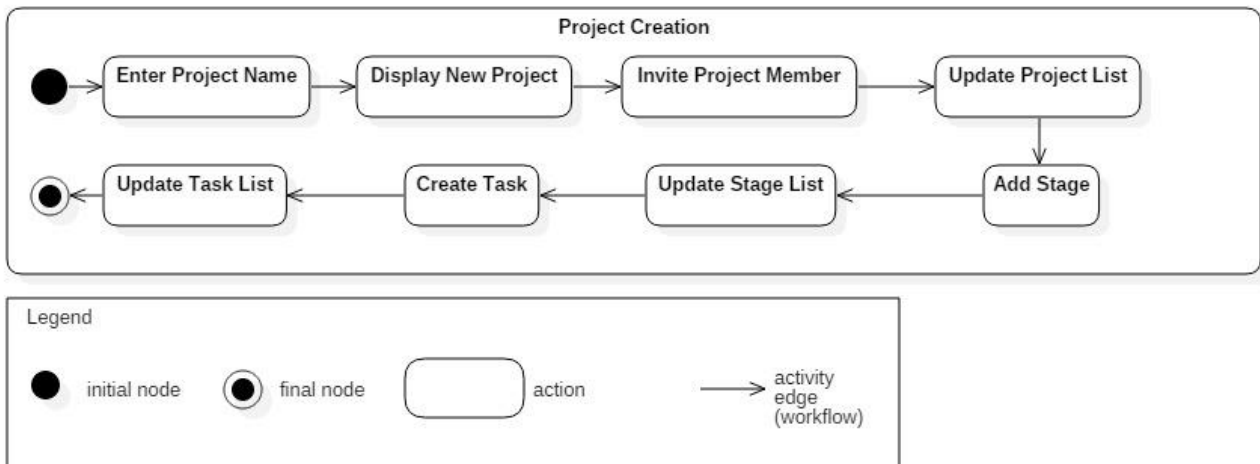


Fig. 9 The coordination model of Project Creation at the agent interaction level

receive this message?

- 3) Is there any implicit message that is not described in the scenario?

The answer to the first question provides information for the message sort. If the stakeholders cannot identify the sort, it can be treated as a normal send message and refine it in next iterations. The answer to the second question provides information for the condition property. The answer to the third question can help to elicit implicit interaction requirements that are not described in the scenario. For example, the requirements engineer captures an implicit message “send notification” from COS to project members, although this message does not explicitly described in the scenario. Then the requirements engineer can add this message to the original scenario.

3. Number the above messages in a temporal order. If the messaging order cannot be predictable until the runtime, the requirements engineer needs to build more than one communication model for the scenario. For example, in a meeting schedule system, the meeting participants can either send their available dates at the same time or one by one. Then one model can be created for the situation that the meeting participants send available dates at the same time, and the other model can be created for the situation that the meeting participants send available dates one by one. The sequence expression [41] is used to represent a lot of knowledge, for instance, the sequence number, concurrency, and the number of message instances. If a message is a concurrent message, we use $*||$ notation to specify concurrent (parallel) execution of messages. If a message is a sequential message, we use $*$

iteration notation to specify the iteration will be executed sequentially. For example, the notation $*||[k:1..2]$ means that there are two messages will be sent concurrently.

Fig. 10 shows a communication model of COS represented using UML 2.0 Communication Diagram. Message with sequence 1.1 denotes the first message within activation 1. Message with sequence 2 follows message with sequence 1, and 2.1 follows 2. Messages 2.2a and 2.2b concurrently follow message 2.1 within activation 2. In this communication model, the “sendNotification(invitation)” message is an implicit message.

5.6 Create Connection Model (Step 4.3)

The heuristics for creating Connection Model at this level are similar to those at the context level. The only difference is that at the agent interaction level, the requirements engineer asks several questions to acquire the domain knowledge for a specific scenario – “Project Creation”, rather than for the whole system.

In the connection model at this level (see Fig. 11), there are also three agents, which are the project owner, COS and project member. The project owner depends on the project member to participate in the project and to provide the personal information. The project owner and COS have dependencies as follows: the COS depends on the project owner to provide a set of resources such as the project name, stages, tasks and invitations. There are also dependencies between the COS and the project member. For example, the project member depends on the COS to provide the invitation notifications. The COS also depends

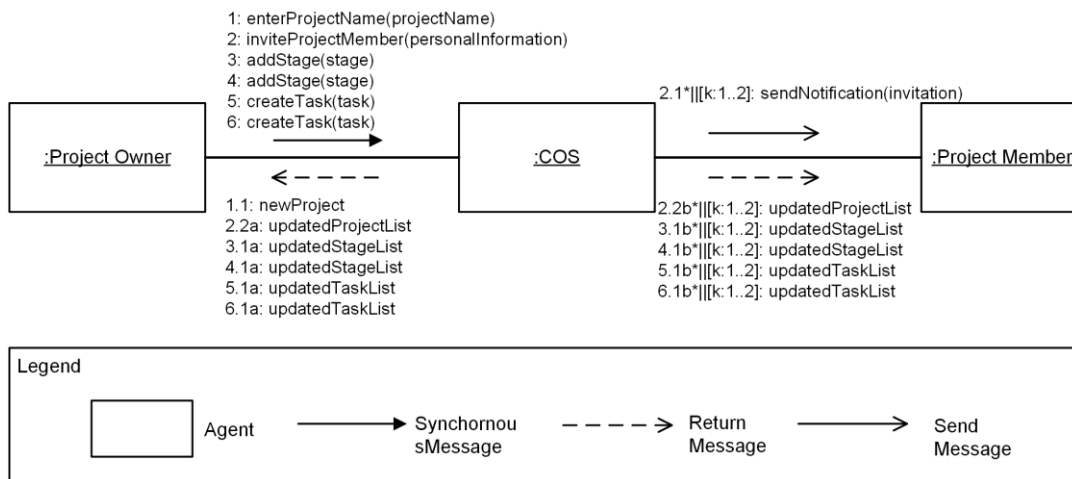


Fig. 10 The communication model of Project Creation at agent interaction level

on the project member to provide the stages.

5.7 Create Collaboration Model (Step 4.4)

In this step the requirements engineer will mostly make use of the answers to question 1), 2), 3) and 7) in Section 5.3 to create Collaboration Model by following the heuristics below:

1. Analyze the high-level goals of each agent in this scenario and map them to the root goals in the collaboration model (Fig. 12). For example, the goals of the project member and COS in this scenario is to add a new project successfully into the boards, i.e. "Project Be Added". If a goal appears more than once in different agents, it means that the goal is not an individual goal but a collaboration goal, such as

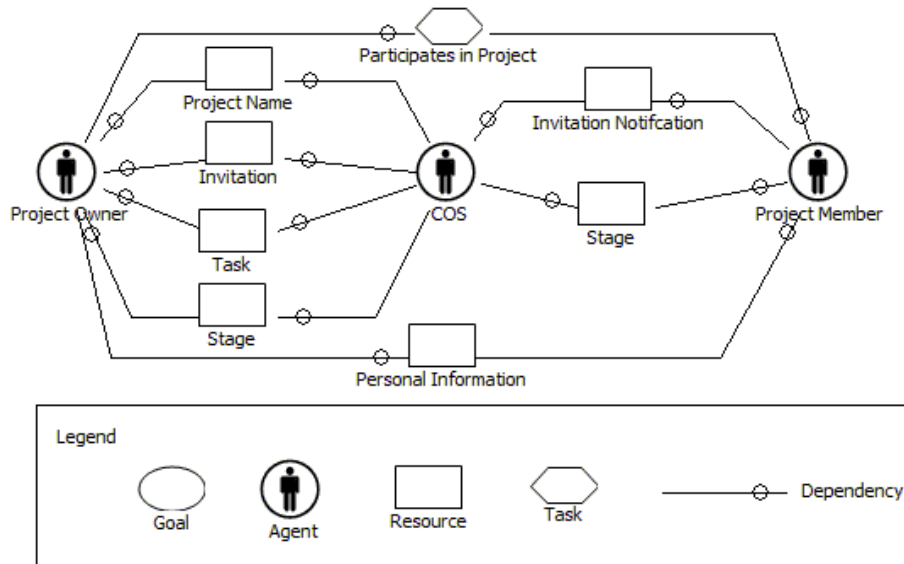


Fig. 11 The connection model of Project Creation at the agent interaction level

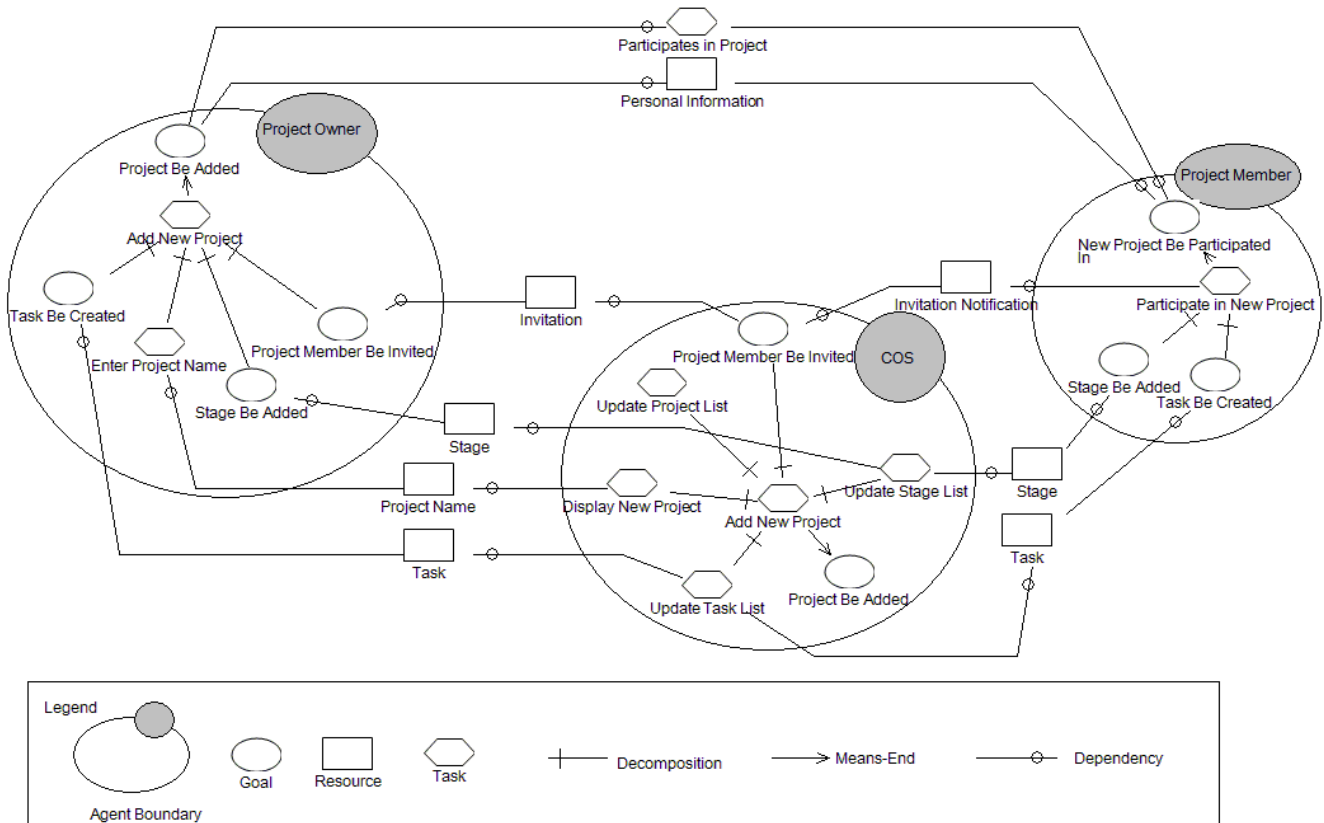


Fig. 12 The collaboration model of Project Creation at the agent interaction level

“Project Be Added”. The goal for the project members is to participate in the new project, i.e. “New Project Be Participated In”.

2. Identify the subgoals and tasks of each agent for the root goals, and structure the subgoals and tasks with the decomposition and means-end relationship. For example, to realize the “Project Be Added” goal, the project owner needs to perform “Enter Project Name” and realize the subgoals “Task Be Created”, “Stage Be Added” and “Project Member Be Invited”.
3. Identify the dependencies for each tasks and subgoals among different agents. For example, the COS has a subtask called “Display New Project”, which depends on the project name entered by the project owner.

The validation of the four-perspective models are required. When the project members checked the collaboration model, they found that they have multiple ways to participate in a new project, such as adding stages, and creating tasks. But in the scenario description, only “Add Stage” is mentioned. Therefore, the requirements engineer can complement the scenario by adding new information to the goal “Task Be Added”. At this point the requirements engineer is ready to move to the agent internal level.

5.8 Define agent internal scenarios (Step 5)

Each leaf goal in Collaboration Model at the agent interaction level suggests a new scenario at the agent internal level. For example, the “Stage Be Added” goal in Fig. 12 derives a new scenario *Add Stage* at the agent internal level. Then we can define this scenario with a statement.

The questions at this level put more emphasis on the internal operations and the condition to perform the operations. Therefore the following questions are asked:

- 1) What operations do you need to do to achieve the goal?
- 2) Is there any condition or constraint for you to perform the operation?
- 3) What are the relationships between these operations?

By answering these questions, the requirements engineer writes the *Add Stage* scenario as follows:

“The project owner enters the stage name of a project and selects to save the new stage to COS. This scenario requires that the project owner logs COS and at least a

project has been created.”

5.9 Model agent internal scenarios (Step 6)

The heuristics for creating Coordination Model and Collaboration Model at this level is the same as those at the agent interaction level. We will not repeat the steps in this paper. In the coordination model (Fig. 13), the *Add Stage* process is divided into two actions: “Enter Stage Name” and “Save New Stage”. To construct the collaboration model (Fig. 14), the requirements engineer has more questions to ask stakeholders, which are similar to those asked at the middle level:

- 1) On whom and what do you depend to perform this operation?
- 2) Do you need to provide the outcome to other agents?

For example, in Fig. 14, to realize the “Stage Be Added” goal, the project owner has to perform two operations that correspond to the actions in the coordination model. The COS depends on the project owner to provide the stage name. Similar to the levels above, the coordination models and collaboration models at this level should also be validated by the requirements engineers.

By applying our approach to the COS, the requirements engineer starts from a contextual scenario and elicit COS’s requirements incrementally. Finally a system of scenarios and requirements models are elicited. At the top level, a contextual scenario and a connection model are obtained; at the middle level, a set of agent interaction scenarios (e.g., *Project Creation*), their corresponding coordination models,

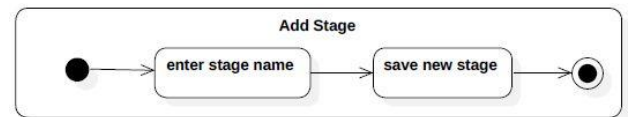


Fig. 13 The coordination model of Add Stage at the agent internal level

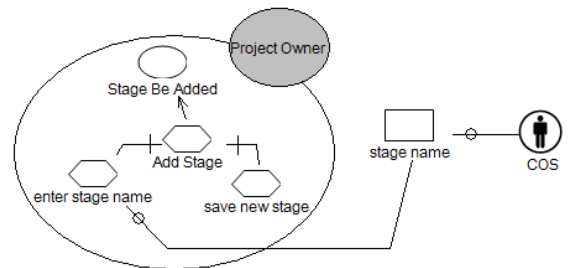


Fig. 14 The collaboration model of Add Stage at the agent internal level

connection models, communication models and collaboration models are defined; at the bottom level, a set of agent internal scenarios (e.g., *Add Stage*), their corresponding coordination models and collaboration models are defined.

6 Related work

In this section, we compare our approach with some closely related approaches by focusing on each other's similarities and differences. We ask these two questions: 1) How are the 4Cs characteristics of e-collaboration systems supported by the existing approaches? 2) How does our approach differ from similar approaches?

6.1 Supporting the 4Cs characteristics

Supported by e-collaboration platforms and tools [10], such as RM-Tool [32] and GroupSystems [55], most current work on requirements elicitation has been focused on improving the elicitation process itself. By contrast, work on how to elicit requirements for e-collaboration systems has so far received little attention. Although several approaches are available for dealing with the elicitation activities concerning the applications in the context of collaborative environments, they do not support all of the 4Cs characteristics. This sub-section discusses some of the closely related approaches to ours.

1. Workflow-oriented elicitation

Coordination is a core concept in e-collaboration systems. Action-Workflow is an early step to capture the coordinative requirements among agents [37]. It defines a basic communication unit called customer-performer in terms of a four-phased loop, including proposing, agreeing, performing, and accepting. However, Action-Workflow loop was not designed towards e-collaboration systems. It only addresses the coordination characteristic, while the intentionality properties of agents are ignored [65]. Therefore there is no support for the connection and collaboration aspect. Similar to Action-Workflow, Fuks et al. [19] provided a common language for representing the collaboration process and functional aspects of a workgroup and guiding the functional specification of e-collaboration systems. Chebbi et al. [7] emphasized the role of workflow playing in inter-organizational

cooperative systems and provided a workflow modeling approach to elicit requirements of workflows and resources among agents, which consists of three steps: workflow advertisement, workflow interconnection and workflow cooperation. Nevertheless, all the above approaches address the interactions and coordination between stakeholders, but none of them take stakeholders' intentions into consideration.

2. Agent-oriented elicitation

To overcome the difficulties of workflow-based approaches in capturing the connection and collaboration requirements, agent orientation was emerging as a new paradigm [65]. Yu [66] argued for six properties for agents, i.e., intentionality, autonomy, sociality, identify and boundary, strategic reflectivity and rational self-interest, and proposed the *i** framework to elicit user requirements in an agent-oriented modeling paradigm. The *i** framework is not only considered as a requirements modeling language, but also a requirements elicitation tool. It uses the SD model and SR model to address the six properties of agents. Although the *i** framework is not designed peculiar to e-collaboration systems, stakeholders in e-collaboration systems share the same properties as agents. So the SD model can support the connection characteristic whereas the SR model can support the collaboration characteristic of e-collaboration systems. However, the descriptions of the activities and actions of e-collaboration systems are not given in *i**. Thereafter, the *i** framework is unable to portray the coordination aspect of e-collaboration systems. In order to overcome the limitations of the *i** framework, there are several requirements elicitation and modeling techniques for cooperative systems to enrich *i**. For example, an extended *i** approach, CSRML (Collaborative Systems Requirements Modeling Language) [52], attempts to provide support for the collaboration, communication and coordination aspects of collaborative systems. Though CSRML categorized tasks into the coordination, the collaboration and communication tasks, CSRML is only capable of describing the structural knowledge, rather than the dynamic knowledge. Another agent-oriented technique is HOMER [62], a Human-Oriented Method for Eliciting Requirements. HOMER uses organizational metaphor to help requirements engineers to elicit requirements by answering a set of questions around the metaphor, such as "What is the purpose of the position?" and "What tasks will be commonly be required?". HOMER can also be integrated into agent-oriented software engineering (AOSE)

frameworks. However, the coordination, communication and connection aspects are not concerned with HOMER. To overcome the problem that *i** does not address the inter-agent communication, Sutcliffe [51] proposed a coupling analysis method to capture and analyze the requirements for social-technical systems or cooperative systems. The coupling analysis method builds on the *i** models and extends *i** by modeling the communications between agents by discourse act types. The coupling analysis method consists of three stages: 1) use cases and system modeling by the *i** models; 2) communication and task analysis; 3) dynamic analysis. Although this method focuses on the requirements analysis such as the agents' workload analysis, it can also be used to elicit the implicit communication requirements. Yet, the coordination aspect is not addressed by this method. While our multi-perspective approach bears close relations to agent-oriented elicitation, it also addresses the coordination and communication aspect of e-collaboration systems.

3. Activity Theory-based elicitation

Activity Theory (AT) of social science also provides support for the identification of contradictory requirements in some AOSE methodologies, such as INGENIAS [42] and Tropos [5]. Requirements Elicitation Guide (REG) based on AT [17] [18] is another attempt to support developers in gathering requirements about the social environment of cooperative applications. The REG technique is developed as a set of tools by incorporating the AT concepts such as areas, aspects and questions to the UML diagrams. A question has a related set of answers that represent possible requirements elicited by that question. By answering a set of questions, developers can follow a generic guide to elaborate the social features that should be considered in cooperative systems. The REG technique contains many of the concepts needed for e-collaboration systems and considers more on the mutual influences between the envisioned system and the human context, but does not deal with issues of agents' coordination and communication.

4. Scenario-based elicitation

Scenarios are widely recognized as an important technique to elicit user requirements for various types of systems. Lau [33] identifies two reasons for the difficulties in requirements elicitation for e-collaboration systems: 1) insufficient engagement of the 'potential' users in requirements capture; 2) insufficient understanding of how RE tools can help people working more effectively with

each other. Lau shared the experience gained from one empirical study in using scenario-driven techniques at the requirements elicitation stage of developing e-collaboration systems. By walking through the scenarios, a shared understanding of the vision could be achieved among stakeholders; and moreover, scenarios can separate a complex process into individual tasks, each of which can be intertwined with the use of IT solutions. In so doing, the second aforementioned problem is tackled. However, Lau's approach was just exploring the possibilities of applying scenarios into requirements elicitation for e-collaboration systems, but not explicitly addressing the 4Cs characteristics. Our approach also starts from a scenario description and makes use of scenarios to help with the elicitation process. Coordination Model also divides an entire business process into a coherent set of tasks (i.e., actions in our approach). Thereafter, the binding of tasks and IT solutions is also enabled in our approach.

Also acknowledged the crucial role played by scenarios in eliciting e-collaboration requirements, Oliveira and Cysneiros [39] adopted scenarios into requirements elicitation in the context of collaborative settings to address aspects such as connection, autonomy, collaboration and pro-activeness. They developed Strategic Dependency Situations (SDsituations) as a simple technique for helping requirements elicitation. They first defined SDsituations, then defined a scenario for each SDsituation and finally used *i** models to organize different scenarios. Our approach also uses *i** to model the intentionality of agents in different scenarios. Besides, they presented the mapping from the concept – *Episode* – in scenarios to the task concept in the *i** model, which is similar to our approach. The episode concept was proposed by Leite et al. [35], which can be mapped to the path in our scenario metamodel (Fig. 6). In addition, Leite et al. defined a scenario consisting of the context, title, goal, resources, actors, episodes and exceptions, some of which can also be found in our scenario metamodel, such as title, goal, and resources. The other elements such as context, actors and exceptions can be mapped to the elements in our scenario metamodel. For example, the actor can be mapped to the agent, while the context and exceptions can be mapped to the source and paths. While our approach does not addresses NFRs, both of the above approaches discussed how to elicit NFRs for collaborative systems by incorporating the constraints into scenarios.

Another approach applying scenarios to requirements

elicitation for e-collaborative systems is MAMIE (from MAcro to MIcro level requirements Elicitation) [4]. MAMIE was developed for eliciting requirements for inter-company collaborative information system (ICIS). MAMIE integrates goals, scenarios and viewpoints within a requirements elicitation approach. In MAMIE, the macro level is used to find goals and use cases, the medium level is based on scenarios, and the micro level is used to discover low-level requirements based on viewpoints. Besides, the relevant NFRs of the system-to-be are coped with at the macro level. Moreover, MAMIE takes the stakeholders' situation into consideration and selects the most suitable elicitation technique for the stakeholder. The selection process is based on Felder-Silverman's Learning Styles Model (LSM) classification [4]. In LSM, people are classified into four categories: Sensing / Intuitive, Visual / Verbal, Active / Reflective, Sequential / Global. Likewise, our multi-perspective modeling approach can satisfy all of the above categories of stakeholders to elicit requirements. For example, Coordination Model and Communication Model are suitable for sensing people to learn facts, whereas Connection and Collaboration Model is suitable for intuitive people to discover possibilities and relationships. The diagrammatic models are suitable for visual people and reflective people that prefer to think according to what they see, whereas the narrative scenarios are suitable for verbal people that prefer explanations and active people that remember things when they put things down. Coordination Model and Communication Model are also suitable for sequential people to understand the requirements easier when following a step-by-step procedure, whereas Connection Model and Collaboration Model is suitable for global people to get the rough features and to find connections. MAMIE provides a systematic guidance to instruct requirements engineers to elicit requirements in an ICIS, but it does not explicitly describe the relationships between the stakeholders and does not encompass the concept of activities, which are crucial for a cooperative information system. In contrast, our approach uses Connection Model to represent the relationships between stakeholders and uses Coordination Model to describe activities.

5. Viewpoint-based elicitation

Viewpoints [38] have long been used to capture different perspectives of user requirements. But the viewpoint method does not explicitly support the elicitation of interactions and the cooperation between actors, or the

activities and actions of e-collaboration systems. In order to overcome the weakness of viewpoints, Kessi et al. [27] proposed VpCIS (Viewpoints for requirement engineering in a Cooperative Information System) to identify user requirements for collaborative information systems from three key aspects, including the organizational aspect, the functional aspect, and the informational aspect. VpCIS incorporates the concept of activities, actors, and group of actors into the existing viewpoint template. However, VpCIS cannot capture dynamic interactions between agents and coordination between actions. Moreover, while VpCIS focuses on the requirements descriptions, it does not show us how to elicit requirements with VpCIS in detail.

6. Other elicitation techniques

In addition to the approaches discussed above, there are other requirements elicitation techniques for collaborative systems, such as questionnaire and interview [2], observations [36], simulation-based elicitation [49]. Alreshidi et al. [2] adopted a structured questionnaire and semi-structured interviews to capture the requirements for collaborative governance solutions – Building Information Modeling (BIM) governance. The questionnaire design concerns the connection, collaboration and communication among stakeholders. Machado et al. [36] developed an observation conceptual model (OCM) and a collaborative observation method aimed at improving the requirements elicitation activity for systems delivered at collaborative workplaces. The OCM is a framework that captures and specifies different dimensions of knowledge through the observations of requirements engineers, such as “what to observe”, “how to observe”, “how to analyze” and “how to represent”. The collaborative observation method defines an iterative elicitation process by preparing, capturing, analyzing, evaluating and representing domain knowledge from various types of materials. After the observation process, the functionalities, the responsibilities and activities of each role can be captured. However, the requirements elicited by this method are very fragmented and do not explicitly support the 4Cs characteristics. Simulation is another emerging technique to elicit requirements for virtual collaborative systems. Silva and Hirata [49] captured the interaction requirements of potential users in collaborative environments emphasizing CIB (Collaborative Information Behavior) activities by simulating the activities among actors that make up CIB. The process of this approach is composed of four steps: define the system, its environment, the restrictions and

permissions of CIB activities, plan the simulation, execute, monitor and control the simulation, analyze and identify requirements. The simulation-based approach addresses the coordination, communication, collaboration aspect of e-collaboration systems as well as the awareness property of agents, while the connection aspect is not supported.

6.2 Model-driven requirements elicitation

Model-driven requirements elicitation [22] is based on the concept of model transformation and refinement. This sub-section discusses some of the well-known model-driven requirements elicitation approaches. Goal modeling is the most commonly established requirements elicitation technique for the early phase of RE [22]. The KAOS framework [56] provides a model-driven way to goal-oriented requirements acquisition. In KAOS, the requirements engineers first identify high-level goals, which is similar to our approach. Then the goals are formalized and objects are identified and enriched by the goals. Next new specific goals are elicited by asking the WHY and HOW question. The agent is also an essential construct in KAOS as each goal can be assigned to an agent. According to Yu [65], KAOS “offers a strong foundation for goal-based reasoning and analysis in agent-oriented modelling”, yet, agents in KAOS do not have rich social relationships and interact non-intentionally. Therefore we cannot only use KAOS to capture the requirements in e-collaboration systems. But the goal-model-driven elicitation steps in KAOS framework provides a lot of guidelines for the requirements elicitation process presented in this paper. Rolland et al. [46] proposed a goal-scenario coupling approach to elicit system requirements at different levels of abstraction. According to Rolland et al., there exists a refinement relationship between the hierarchy of scenarios. The refinement relationship relates a higher-level scenario to a lower-level scenario. Under this relationship, higher-level (less detailed) scenarios are refined into lower-level (more detailed) scenarios by a goal-model-driven process. The goal-scenario coupling approach suggests an interactive activity between scenario transformation and scenario writing: new scenarios discovered by the current round of transformation needs to be written up before the next round of transformation can be performed again. The approach presented in this paper is also an interactive process of scenario authoring and goal discovering. The difference

between the two is that Rolland et al.’s process focuses on the goal model transformation only, whereas our process transforms a scenario into four different models, which include the goal model. Since our approach is towards e-collaboration systems, we emphasize the connection, coordination and communication between agents in addition to the intentionality of each agents. Besides, unlike Rolland et al., we consider in this paper the scenario authoring process. We design a set of questions to help the requirements engineers to elicit the knowledge that is needed for a scenario.

Georg et al. [20] proposed a methodology that combines the Activity Theory (AT), User Requirements Notation (URN) goal and scenario modeling (AT/URN) to elicit, analyze and evolve requirements. It first performs the AT analysis to generate the ASD (Activity System Diagram) network models, then transform it to GRL (Goal-oriented Requirement Language) models, and finally create the UCM (Use Case Maps) to design scenario models based on the goal models. In the integrated AT/URN approach, while AT provides a broad, conceptual framework for exploring social requirements, such as human activities and their mediating relations, URN provides formality to analyze and exploit traceability. Like the AT/URN approach, the approach presented in this paper also combines the use of goal models and scenarios. However, while the AT/URN approach put emphasis on how to define and evolve social requirements at one level, our approach focuses on the iterative requirements elicitation at multiple levels of abstraction and from multiple perspectives.

The i^* modeling framework introduces an intentional and social ontology for requirements engineering. There are several model-driven elicitation approaches based on the i^* framework. For example, Cysneiros and Yu [12] proposed a model-driven requirements engineering methodology that builds on the i^* framework for systems in a collaborative environment. This methodology starts from the LEL (Language Extended Lexicon) and uses the strategic actor as the central construct to elicit and organize requirements, which is composed of five basic steps: 1) build a lexicon to obtain a vocabulary about the domain, 2) build a social structure of strategic actors to derive the initial SD models, 3) build a first-cut dependency model to derive more complete SD models, 4) add intentionality for each actor and build the SR models, and 5) reason about the alternative solutions. The SD model and SR model are the core to drive the elicitation process. Oliveira and Cysneiros

[39] integrated the LEL approach, scenarios technique and i* framework to elicit requirements. Their approach is divided into four steps: elaborate LEL to obtain the general domain knowledge, define SDsituations, define scenarios and model intentionality with i*. The development of the i* models may demand changes in SDsituations as well as in the definition of scenarios, which results in the enhancement of original requirements. The approach presented in this paper is similar to the above two approaches in that all of the three approaches address the agent characteristics such as such as autonomy, sociality, and intentionality and select i*/scenarios coupling approach to capture these aspects. Our approach differs from the two approaches in that the former is mainly a top-down elicitation process that starts from a contextual scenario and refines the high-level requirements into more detailed ones by coupling scenarios and the four-perspective models, whereas the latter ones start from the lexicon and refine the coarse-grained requirements by encoding more information to the i* models step by step, which builds a traceability link between the lexicon and the i* models. Compared to the two approaches, our approach put more emphasis on the hierarchy of user requirements.

In addition to goal-model-driven elicitation and i*-model-driven elicitation, there are some other elicitation work driven by organizational models. For example, the EKD (Enterprise Knowledge Development) approach [6] to requirement engineering is driven by a set of organizational models, the goal model, the concepts model, the business rules model, the business process model, the actors and resources model, and the technical components and requirements model. Oliveira et al. proposed a REMO-EKD (Requirements Elicitation oriented by business process MOdeling for EKD) [40] approach in order to support the elicitation of software requirements based on EKD models. The REMO-EKD technique supports the elicitation of FRs and NFRs as well as business rules. REMO-EKD consists of a set of ten heuristics, each of which is composed of: 1) a set of components of the EKD model it is related to, 2) the description of the heuristic, and 3) the guideline about how to elicit the requirements. However, these approaches mostly tell us what to elicit, while they pay little attention on how to elicit requirements. Besides, both the EKD and REMO-EKD approaches have limitations in dealing with agents' sociality.

6.3 Summary

Based on the above comparison, we can now answer the two questions raised in the beginning of the section.

First, *how are the 4Cs characteristics of e-collaboration systems supported by the existing approaches?* As Table 3 shows, none of the approaches supports all of the 4Cs. Our work is therefore timely in addressing this gap.

Second, *in what way does our approach differ from similar approaches?* Our approach is not only a scenario-based technique but also a model-driven elicitation process. In comparison with similar scenario-based approaches, our approach covers the 4Cs of e-collaboration systems, and moreover, our approach structures different levels of scenarios into a consolidated hierarchy. In comparison with other model-driven elicitation approaches, our approach performs the elicitation in a stepwise refinement fashion, from the contextual level to the agent interaction to agent internal.

In comparison with some elicitation approaches such as the i* framework [64], CSRML [52], SDsituations [39], our approach currently only focuses on the elicitation of FRs. According to Leite et al. [35] and Oliveira and Cysneiros [39], a scenario is bounded by a *context*, which is constrained by a property called *constraint*. In addition, *resources* and *episodes* (i.e., a sequence of actions in our approach) also have constraints, which derive NFRs. Likewise, we can add the constraint property to the scenario, resource, action in our approach, and integrate them into four-perspective models respectively. The scenario notation proposed in [39] can be adopted to format e-collaboration scenarios into a more structural way. Cysneiros and Leite [11] also proposed a process to elicit NFRs and trace them to UML models. The i* modeling framework provides support for NFRs through softgoals [64]. There are two ways to integrate NFRs to Coordination Model and Communication Model. The first way is to create new symbol for NFRs as an attachment to the existing elements [11], while the second way is to treat NFRs as a constraint attribute of the existing elements and RE-Tools provides the functionality [50]. Both ways are recommended.

7 Practical implications

In this section, we discuss the practical aspects of our approach. We base our discussion on an industrial study

conducted by Knauss et al. [28]. The study identified four RE challenges for collaborative industrial applications. In what follows, we first discuss how our approach can address these RE challenges and then show the relevance of our approach to industrial applications.

7.1 Addressing RE challenges

Challenge 1: The management of domain and technical knowledge across all organizational levels and actors.

As Knauss et al. reported [28], three scope levels are identified to analyze collaborative systems. Scope level 1 presents the system's relationships to other systems and an external view on the system. Scope level 2 analyzes the relationships within the system by managing the context and mapping requirements to actors. Scope level 3 shows the system from the perspective of a single organization.

Our approach also proposes three levels, i.e., the context level, agent interaction level and agent internal level. The context level can be mapped to Scope level 2 whereas the agent interaction level and agent internal level can be mapped to Scope level 3. Although we don't discuss Scope

level 1 in our approach, Connection Model can also be used at Scope level 1 to show the inter-relationships among different systems. Besides, in our approach, domain knowledge can be captured, described and modeled at three levels, and technical knowledge can be managed at the agent internal level as Collaboration Model used at this level in essence has the ability to capture and model technical solutions by using the means-end and decomposition relationship.

Challenge 2: The management of stakeholder interaction across multiple organizational boundaries and between teams.

According to Knauss et al. [28], practitioners feel a very high need for interactions both globally and locally in capturing and communicating requirements. In our approach, the interaction of stakeholders is addressed from different aspects. Connection Model is used to outline each agent's boundary and capture the strategic relationships between different agents. Besides, we designed a set of questions to promote stakeholder interaction. For example, question 1), 2), 3) in Section 5.2 at the context level and

Table 3 A summary of requirements elicitation techniques for collaborative systems concerning the 4Cs characteristics (+ notation applied; - notation not applied)

Elicitation technique	Category	Collaboration	Connection	Communication	Coordination
Action-workflow loop [37]	Workflow-oriented	-	-	-	+
3c collaboration model [19]	elicitation	-	-	+	+
Workflow modeling [7]		-	-	+	+
i* [64][65]	Agent-oriented	+	+	-	-
Coupling analysis [51]	elicitation	+	+	+	-
CSRML [52]		+	+	-	-
HOMER [62]		-	+	-	-
REG [17][18]	AT-based elicitation	+	+	-	-
Lau's approach [33]	Scenario-based	-	-	-	-
SDsituations [39]	elicitation	+	+	-	+
MAMIE [4]		+	-	+	-
VpCIS [27]	Viewpoint-based elicitation	-	+	-	-
Collaborative observation method [36]	Observation	-	-	-	-
Questionnaire and interview [2]	Questionnaire and interview	+	+	+	-
Silva and Hirata's approach [49]	Simulation	+	-	+	+

question 3), 5), 7) in Section 5.3 at the agent interaction level will motivate the stakeholders to reflect whether they have relationships with others and then drive them to interact with other stakeholders.

Challenge 3: The systematic transformation from the requirements flows into technological and strategic decisions.

Knauss et al. [28] suggested that the transformation should rely on two general flows of requirements, i.e. the top-down requirements flow and bottom-up requirements flow. According to the observation of current practice, most practitioners like to develop scenarios in a top down manner [46], which is effective and natural. Yet, the difficulty with this manner is in the control of the top down decomposition [28].

We have seen in Section 5 of the paper that our approach uses a system of scenarios and models to tackle this problem. The contextual scenario, agent interaction scenarios and agent internal scenarios refine requirements in a controllable process. However, as mentioned in [28], only top-down requirements flow is not sufficient to address this challenge. Therefore, we make use of the feedbacks from stakeholders (Fig. 7) as the bottom-up requirements flows to transform new knowledge into emergent requirements. The refinement process is therefore interactive by combining the top-down and bottom-up requirements flows. Moreover, our approach offers two models – Connection Model and Collaboration Model – to help with the tradeoffs of technological and strategic decisions. For example, it was reported that in many projects [63][65][68], the SR model and SD model contribute to elicit technical solutions.

Challenge 4: The early involvement of stakeholders.

Knauss et al. [28] suggested that early involvement of stakeholders in requirements elicitation is crucial in ensuring the success of collaborative systems. Our approach addresses this challenge by asking the stakeholders a set of questions about the system context. Our approach also actively involves the participation of the stakeholders in the definitions of different levels of scenarios and the design of requirements models.

7.2 Relevance to industrial applications

The study reported by Knauss et al. [28] is based on IBM CLM tool suite [25], which is a collaborative platform. CLM supports a host of IBM tools, including IBM Rational Team Concert (RTC), Rational Quality Manager (RQM) and Rational Requirements Composer (RRC). CLM delivers requirements management, quality management, change and configuration management and project planning and tracking. In this sub-section, we will use CLM to demonstrate the relevance of our proposed approach to industrial applications. Due to space limitation, here we have only selected one scenario for each level to illustrate how our approach works in CLM, as shown in Fig. 15.

We first use Connection Model to describe the Scope level 1 of IBM CLM system (the top-left part of Fig. 15), which focuses on the features such as target market, the participants and so on. Then we map the context level to Scope level 2 (the top-right part of Fig. 15) and derive the contextual scenario, which is further modeled by Connection Model again to analyze the relationships of different agents in CLM. The agent interaction level and agent internal level are mapped to Scope level 3, which focuses on the requirements refinement using the multi-perspective modeling approach. At the agent interaction level, a system of agent interaction scenarios are derived and then transformed into the four-perspective models. In the middle of Fig. 15, the four-perspective models for the *Test Case Management* scenario at the agent interaction level are shown. At the agent internal level, a system of agent internal scenarios are derived and then transformed into the coordination models and collaboration models. At the bottom of Fig. 15, the coordination model and collaboration model for the *Test Case Creation* scenario are shown.

From the above discussion, we can see that the multi-perspective modeling approach has the potential to apply in the industry.

8 Conclusion

Based on the 4Cs characteristics of e-collaboration systems, this paper has proposed a four-perspective modeling approach for eliciting user requirements of e-collaboration systems. Each perspective model focuses on one aspect of the e-collaboration process and contributes to one viewpoint of a total requirement specification. Collectively, these models provide a comprehensive view of

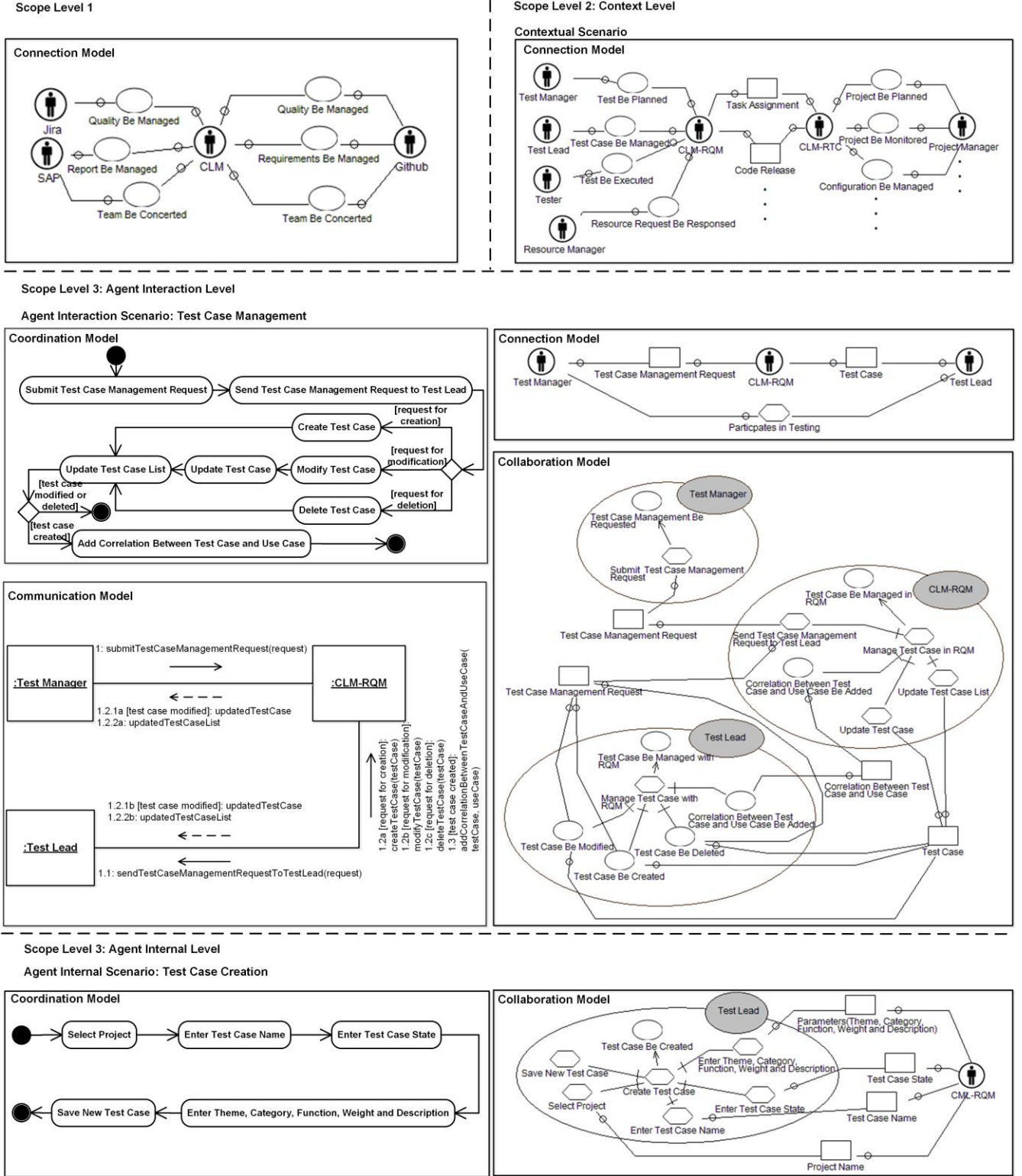


Fig. 15 The multi-perspective modeling approach applied on IBM CLM tool suite.

requirements for e-collaboration systems. These models, together with the modeling process, offer requirements engineers a tool to elicit, analyze and represent user requirements in a systematic way.

To evaluate our approach, we have used the solution proposal evaluation criteria suggested by Wieringa et al. [61]. Specifically, we have:

1. clearly explained the problem to be solved by the

proposed approach.

2. demonstrated the novelty of the proposed approach by comparing it with similar approaches.
3. described the approach in such a detail that the approach should be replicated by other researchers.
4. illustrated the feasibility of our approach through a real world e-collaboration example.
5. discussed the practical implications of our approach

based on an industrial study.

6. compared related work in detail by identifying the similarities and differences between our approach and the existing approaches.

The key strength of our approach is that we start from the four characteristics of e-collaboration systems and explore four distinct models in eliciting requirements for e-collaboration systems. We found that there is an intrinsic relationship between these characteristics and the four-perspective models.

The main limitation of the proposed approach is that creating four models at multiple levels is clearly time consuming. However, in practice, not every scenario contains all of the 4Cs and hence normally the number of models for each scenario is much smaller than what has been illustrated in this paper. There are two possible ways to solve this problem. One is to integrate the multi-perspective modeling approach with Agile development [8]. The requirements engineers do not need to create 100% accurate and detailed models for all scenarios at the first time; instead, they can create sufficiently accurate and detailed models for different scenarios with the change of iterations. For example, some scenarios are the key requirements to the e-collaboration system, then in the first several iterations, we can create models of the three levels for these key scenarios, while other less important scenarios can be elicited just at the contextual level or agent interaction level in these iterations and refined in the following iterations. In other words, we can create some “Just In Time” (JIT) models in the requirements elicitation process. Another way is to automate the text-to-model transformation. For example, by employing natural language processing tools, we can transform the coordination model from the scenario through two basic steps: the syntax-level analysis and text-level analysis. The syntax-level analysis is obtain the grammatical structure of each requirement and extract the necessary actions from each requirement, whereas the text-level analysis is to analyze the relationship among requirements and extract the workflows among actions [58].

For complex scenarios, our suggestion is to decompose them into a set of use cases before applying our modeling approach. Use cases are more structural and simpler than scenarios, which will simplify our approach and hence improve the efficiency of our approach.

Our approach currently does not support model

management, traceability and evolution. We intent to address these issues in future research.

We encourage RE researchers and practitioners to use our approach and any feedback would be greatly appreciated. We hope to conduct an industrial case study to put our approach into practice and to observe its strengths and limitations.

Acknowledgments

We wish to thank the reviewers for their helpful comments and suggestions. This work was supported by the National Natural Science Foundation of China under Grant No. 61402406 and No. 61602412, and is in collaboration between Zhejiang Gongshang University and the University of Manchester.

References

1. Ali H, Macaulay L, Zhao L (2009) A collaboration pattern language for eParticipation: a strategy for reuse. In: ECEG'09, pp 29-38
2. Alreshidi E, Mourshed M, Rezgui Y (2016) Requirements for cloud-based BIM governance solutions to facilitate team collaboration in construction projects. *Requir Eng*: 1-31.
3. Baudin V, Drira K, Villemur T, Tazi S (2004) A model-driven approach for synchronous dynamic collaborative e-learning. *E-Education applications: human factors and innovative approaches*: 44-65
4. Bendjenna H, Zarour NE, Charrel PJ (2010) Eliciting requirements for an inter-company cooperative information system. *J Syst Inf Tech* 12(4): 305-333
5. Bresciani P, Perini A, Giorgini P, Giunchiglia F, Mylopoulos J (2004) Tropos: an agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems* 8(3): 203-236
6. Bubenko JA, Persson A, Stirna J (2001) EKD - Enterprise Knowledge Development User Guide https://people.dsv.su.se/~js/ekd_user_guide.html. Retrieved 2 Jan 2017
7. Chebbi I, Dustdar S, Tata S (2006) The view-based approach to dynamic inter-organizational workflow cooperation. *Data Knowl Eng* 56(2): 139-173
8. Chmuturi M (2013) REM in Agile Projects. In: *Requirements Engineering and Management for Software Development Projects*. Springer, New York, pp 217-232
9. Cook N (2008) *Enterprise 2.0: how social software will*

- change the future of work. Gower Publishing, Brookfield
10. Coughlan J, Macredie RD (2002) Effective communication in requirements elicitation: a comparison of methodologies. *Requir Eng* 7(2): 47-60
 11. Cysneiros LM, Leite JCSDP (2004) Nonfunctional requirements: from elicitation to conceptual models. *IEEE Trans Softw Eng* 30(5), 328-350
 12. Cysneiros LM, Yu ES (2003) Requirements engineering for large-scale multi-agent systems. *Lecture Notes in Computer Science* 2603(4):39-56
 13. Dahlan A, Rahman A, Abdulatif ZK, Mahmoud I, Aydin M (2011) Collaborative healthcare system between clinics and hospitals in Malaysia. *IJCIIS* 2(2): 92-100
 14. Ferrari A, Spoleitini P, Stefania G (2016) Ambiguity and tacit knowledge in requirements elicitation interviews. *Requir Eng* 21(1): 333-355
 15. Franchi E, Poggi A, Tomaiuolo M (2013) Open social networking for online collaboration. *IJeC* 9(3): 50-68
 16. Franch X (2010) Fostering the Adoption of i* by Practitioners: Some Challenges and Research Directions. *Intentional Perspectives on Information Systems Engineering 2010*: 177-193
 17. Fuentes R, Gómez-Sanz, JJ, Pavón J (2004) Towards requirements elicitation in multi-agent systems. *European Meeting on Cybernetics and Systems Research*. In: EMCSR'04
 18. Fuentes, R, Gómez-Sanz JJ, Pavón J (2004) Activity Theory Applied to Requirements Elicitation of Multi-Agent Systems. In: *International Workshop on Activity Theory Based Practical Methods for It-Design*.
 19. Fuks H, Raposo A, Gerosa MA, Pimentel M, Lucena CJ (2007) The 3c collaboration model. In: *Encyclopedia of E-Collaboration*. Information Science Publishing, Hershey, pp 637-644
 20. Georg G, Mussbacher G, Amyot D, Petriu D, Troup L, Lozano-Fuentes S, et al (2015) Synergy between activity theory and goal/scenario modeling for requirements elicitation, analysis, and evolution. *Information & Software Technology* 59(C): 109-135
 21. Haumer P, Pohl K, Weidenhaupt K (1998) Requirements elicitation and validation with real world scenes. *IEEE Trans Softw Eng* 24(12): 1036-1054
 22. He C, Mussbacher G (2016) Model-Driven Engineering and Elicitation Techniques: A Systematic Literature Review. In: *REW'16*
 23. Hickey AM, Davis AM (2003) Elicitation technique selection: how do experts do it?. In *RE'03*, pp 169-178
 24. Hooper JW, Pei H (1982) Scenario-based prototyping for requirements identification. *ACM Sigsoft Software Engineering Notes*, 7(5):88-93
 25. IBM (2017) Rational solution for Collaborative Lifecycle Management. https://www.ibm.com/support/knowledgecenter/en/SS2L6K_6.0.2/com.ibm.help.common.jazz.calm.doc/topics/c_clm_overview.html. Accessed 01 August 2017
 26. Kaindl H (2000) A design process based on a model combining scenarios with goals and functions. *IEEE Trans Systems, Man, and Cybernetics - Part A: Systems and Humans* 30(5): 537-551
 27. Kessi K, Oussalah M, Alimazighi Z (2014) Viewpoints for requirement engineering in a cooperative information system (VpCIS). *New perspectives in information systems and technologies* 1: 299-308
 28. Knauss E, Yussuf A, Blincoe K, Damian D, Knauss A (2016) Continuous clarification and emergent requirements flows in open-commercial software ecosystems. *Requir Eng*: 1-21
 29. Kock N, Davison R, Wazlawick R, Ocker R (2001) E-collaboration: a look at past research and future challenges. *J Syst Inf Tech* 5(1): 1-8
 30. Kock N, Lynn GS (2012) Research article electronic media variety and virtual team performance: the mediating role of task complexity coping mechanisms. *IEEE Trans Profes Comm* 55(4): 325-344
 31. Kulkarni D, Ahmed T, Tripathi A (2012) A generative programming framework for context-aware CSCW applications. *TOSEM* 21(2): 11
 32. Lang M, Duggan J (2001) A tool to support collaborative software requirements management. *Requir Eng* 6(3): 161-172
 33. Lau LMS (2008) Scenarios for e-collaboration are only part of the story. In: *Encyclopedia of E-Collaboration*. Information Science Publishing, Hershey, pp 547-553
 34. Le Dinh T, Rinfret L, Raymond L, Dong Thi BT (2013) Towards the reconciliation of knowledge management and e-collaboration systems. *Interactive Technology and Smart Education* 10(2): 95-115
 35. Leite JCSDP, Hadad GDS, Doorn JH, Kaplan GN (2000) A scenario construction process. *Requir Eng* 5(1): 38-61
 36. Machado RG, Borges MRS, Gomes JO (2008) Supporting the System Requirements Elicitation through Collaborative Observations. *Groupware: Design, Implementation, and Use*. Springer Berlin Heidelberg
 37. Medina-Mora R, Winograd T, Flores R, Flores F (1992) The action workflow approach to workflow management

- technology. In: CSCW'92
38. Nuseibeh B, Kramer J, Finkelstein A (2003) ViewPoints: meaningful relationships are difficult! In: ICSE'03, pp 676-681
 39. Oliveira ADPA, Cysneiros LM (2006) Defining Strategic Dependency Situations in Requirements Elicitation. In: WER'06, pp. 12-23
 40. Oliveira MD, Viana D, Conte T, Vieira S, Marczak S (2013) Evaluating the REMO-EKD technique: A technique for the elicitation of software requirements based on EKD organizational models. In: EmpiRe'13
 41. OMG (2011) 2.4.1 superstructure specification. <http://www.omg.org/spec/UML/2.4.1/Superstructure/PDF>. Accessed 11 May 2016
 42. Pavón J, Gómez-Sanz JJ, Fuentes R (2005) The INGENIAS methodology and tools. In: Agent-Oriented Methodologies. IGI Global, Hershey, pp 236-276
 43. Plihon V, Ralyté J, Benjamin A, Maiden NAM, Sutcliffe A, Dubois E, et al (1998) A reuse-oriented approach for the construction of scenario based methods. <ftp://sunsite.informatik.rwth-aachen.de/pub/CREWS/CREWS-98-22.pdf>. Retrieved 10 June 2016
 44. Riemer K (2010) E-collaboration systems: identification of system classes. In: E-collaboration technologies and organizational performance: current and future trends. IGI Global, Hershey, pp 176
 45. Rolland C, Achour CB, Cauvet C et al (1998) A proposal for a scenario classification framework. *Requir Eng* 3(1): 23–47
 46. Rolland C, Souveyet C, Achour CB (1998). Guiding goal modeling using scenarios. *IEEE Trans Soft Eng* 24(12): 1055–1070
 47. Ross DT (1977) Structured analysis (SA): A language for communicating ideas. *IEEE Trans Softw Eng* 3(1): 16-34
 48. Rutkowski AF, Vogel DR, Van Genuchten M, Bemelmans TM, Favier M (2002) E-collaboration: the reality of virtuality. *IEEE Trans Profes Comm* 45(4): 219-230
 49. Silva APCD, Hirata CM (2012) A simulation-based method for eliciting requirements of online CIB systems. *Lecture Notes in Business Information Processing*, 140:34-52
 50. Supakkul S, Chung L (2012) The RE-Tools: A multi-notational requirements modeling toolkit. In: RE'12, pp 333-334
 51. Sutcliffe A (2001) Requirements engineering for complex collaborative systems. In: RE'01, pp 110-117
 52. Teruel MA, Navarro E, López-Jaquero V, Montero F, González P (2011). CSRML: a goal-oriented approach to model requirements for collaborative systems. In: ER'11, pp 33-46
 53. Treude C, Storey MA (2012) Work item tagging: Communicating concerns in collaborative software development. *IEEE Trans Soft Eng* 38(1): 19-34
 54. Uchitel S, Kramer J, Magee J (2001) Detecting implied scenarios in message sequence chart specifications. *ACM SIGSOFT Soft Eng Notes* 26(5): 74-82
 55. Valacich JS, Dennis AR, Nunamaker JF (1991) Electronic meeting support: the GroupSystems concept. *Int J Man-Machine Studies* 34(2): 261-282
 56. Van Lamsweerde A (2001) Goal-oriented requirements engineering: a guided tour. In: RE'01, pp 249-262
 57. Wang Y (2005) E-collaboration: a literature review. <http://conference.iproms.org/sites/conference.iproms.org/files/PID172674.pdf>. Accessed 10 October 2014
 58. Wang Y, Jiang B, Wang T (2016) Using workflow patterns to modeling and validating service requirements. In: RePa'16
 59. Wang Y, Zhao L, Wang X, Yang X, Supakkul S (2013) PLANT: A pattern language for transforming scenarios into requirements models. *IJHCS* 71(11): 1026-1043
 60. White SA (2004) Introduction to BPMN. http://www.ebm.nl/wp-content/uploads/2010/05/Introduction_to_BPMN.pdf. Retrieved 8 June 2016
 61. Wieringa R, Maiden N, Mead N, Rolland C (2006) Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requir Eng* 11(1): 102-107
 62. Wilmann D, Sterling L (2005) Guiding agent-oriented requirements elicitation: HOMER. In: QSIC'05
 63. Yu ES (1994) Modelling strategic relationships for process reengineering. Doctoral Dissertation, University of Toronto Toronto
 64. Yu ES (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: RE'97, pp 226-235
 65. Yu ES (2001) Agent orientation as a modelling paradigm. *WIRTSCHAFTSINFORMATIK* 43(2): 123-132
 66. Yu ES (2001) Agent-Oriented Modelling: Software versus the World. In: Agent-Oriented Software Engineering II. Springer, Berlin, pp 206-225
 67. Yu ES, Mylopoulos, J, Lesperance Y (1996) Modelling the organization: New concepts and tools for re-engineering. *IEEE Expert*: 16-23
 68. Zowghi D, Coulin C (2005) Requirements elicitation: a survey of techniques, approaches, and tools. In: Engineering and managing software requirements. Springer, Berlin, pp 19-46

Responses to Reviewers' Comments

Based on the review comments and suggestions, we have carried out further research into related work that has been published in REJ and RE Conference series and this has been reflected in the revised manuscript. We have also improved the description of the paper and the quality of the figures. In the following section, we describe how we have addressed your concerns in the revision.

1. Overall Responses to Review Comments

- **Figures:** Fig. 1, Fig. 2, Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 10, Fig. 11, Fig. 12, Fig. 14 and Fig. 15 are revised to improve the quality and readability of these figures.
- **Case study:** We have stated in the conclusion that the real case study will be conducted in the future work.
- **Related work:** We have compared our work to some of closely related work that has been published in REJ and RE Conference series, This work is cited in the manuscript as [2], [28], [32], [35], [45], [51], [61]. Of these references, [2] and [51] are new, added in this revision, to demonstrate our continuous review of related work. Table 3 is also updated.
- **Writing style:** We have reviewed the entire manuscript in detail and corrected several typos and grammatical errors. For example, we changed “in this section” to “in this sub-section” when referring to a sub-section.

2. Response to reviewer 1's comments:

>>>>

While the paper is more polished now, but the main issue of industrial and practical relevance remains. In the opinion of this reviewer it cannot be proved in argumentative way and with a simple example (as currently in section 7). For this to be acceptable to a top journal a real case study needs to be conducted.

<<<<

Answer: Thank you for your comment. We agree with you that the only way to demonstrate the industrial and practical relevance of our work is through a real world case study or application. In this paper, we believe that we have laid down a solid foundation for our proposed approach. In our future work, we hope to conduct real world validation of our approach.

2. Response to reviewer 2's comments:

>>>>

Please check in the paper reference to "all", in particular at page 23 lines 13/14.

At 7.2 it should refer to Sub-Section and not Section (In this section ...).

<<<<

Answer: Thank you for spotting this mistake. We have made appropriate corrections.

>>>>

Please make the Figures easier to read, in special Fig. 15.

<<<<

Answer: We have improved the quality of the following figures: Fig. 1, Fig. 2, Fig. 4, Fig. 5, Fig. 6, Fig. 7, Fig. 8, Fig. 10, Fig. 11, Fig. 12, Fig. 14 and Fig. 15. We hope that the figures are now more readable.

>>>>

Be more specific about the transformation from scenarios to the coordination model (page 24, lines 55/59).

<<<<

Answer: Thank you for the suggestion. We have added the explanation about the transformation from scenarios to the coordination model in the conclusion.