Software Engineering Ontology – the Instance Knowledge (Part II)

Wongthongtham P[†], Chang E[†], Dillon T[†] and Sommerville I[††],

Digital Ecosystems and Business Intelligence Institute, Curtin University, Australia

School of Computer Science, St Andrews University, UK

Summary
Software engineering ontology provides software engineering concepts – what they are, how they are related, and can be related to one another – for representing and communicating over software engineering knowledge and project information through the internet. The ontology enables effective ways of sharing and reusing the knowledge and the project information for remote software engineers and software developers. Reaching a consensus of understanding is of benefit in a distributed multi-site software development environment. Software engineering knowledge is represented in the software engineering ontology whose instantiations are undergoing evolution. Software engineering ontology instantiations signify project information which is shared and has evolved to reflect project development, changes in software requirements or in the design process, to incorporate additional functionality to systems or to allow incremental improvement, etc. This evolution of instances provides many new challenges to an ability to design and deliver project information. In this paper, we present platforms development to facilitate software engineering ontology instantiations management.

Key words:
Ontology, Software Engineering, Multi-site Software Development, Knowledge Engineering.

1. Introduction

The software engineering ontology defines common sharable software engineering knowledge including particular project information [1]. The world's first Software Engineering Ontology is available online at www.seontology.org. Software engineering ontology typically provides software engineering concepts – what they are, how they are related, and can be related to one another – for representing and communicating over software engineering knowledge and project information through the internet [2]. These concepts facilitate common understanding of software engineering project information to all the distributed members of a development team in a multi-site development environment. This should not be confused with the distributed systems, such as CORBA where the development is centralised but deployment is distributed. The ontology enables effective ways of sharing and reusing the knowledge and the project information for remote software engineers and software developers. Reaching a consensus of understanding is of benefit in a distributed multi-site software development environment. Software engineering knowledge is represented in the software engineering ontology whose instantiations are undergoing evolution. Software engineering ontology instantiations signify project information which is shared and has evolved to reflect project development, changes in software requirements or in the design process, to incorporate additional functionality to systems or to allow incremental improvement, etc. This evolution of instances provides many new challenges to an ability to design and deliver project information.

In this paper, we present platforms development to facilitate software engineering ontology instantiations management. The software engineering ontology is made available to any application to deploy. The ability to make use of the software engineering knowledge, described in the software engineering ontology, enables applications in the systems to have capabilities in managing instance knowledge in multi-site distributed software development.

2. Platforms Framework

We illustrate how software engineering ontology facilitates the communication framework and allows knowledge sharing through platforms. This is how the man-machine system interfaces particularly works. The man-machine interactions were designed and developed into four platforms: Knowledge Navigation Platform, Question and Issue Platform, Suggestion and Proposal Platform and Solution Decision Platform.

The Knowledge Navigation Platform, or Navigation Platform for short, is basically for all team members to navigate or query shared domain knowledge and instance knowledge. Anyone can view domain knowledge for classification of certain concepts and can view instance knowledge for clarification of project agreements or understandings. However, permission is needed in the

Navigation Platform if team members want to make changes to instance knowledge. Authorised members can make changes from the Navigation Platform. Unauthorised members can propose changes as an issue. The issue will then be processed through the Question Platform, Suggestion Platform, and Solution Platform depending on how substantial the changes are (minor or major changes).

The Question and Issue Platform, or Question Platform for short, is for raising difficulties encountered in the software development or issues which occurred within the team. A member at any one site logs a project matter through the Question Platform. Basically issues raised would be like comments or debates over project data and project agreement. After all, a question made needs to be elucidated by clarifying the specific project data he/she wants to discuss. Technically, the project data here in the thesis are known as instance knowledge. As a member progresses towards clarifying the issue, the retrieval of relevant instance knowledge by the platform assists other members to have a clear understanding of the issue as well as assisting them to recognise the instance knowledge they have been discussing. This is particularly useful when team members work on many projects simultaneously.

The Suggestion and Proposal Platform, or Suggestion Platform for short, is as its name suggests, for all team members to propose the potential solutions of the discussion or issue proposed in the Question Platform. Thus, this platform mainly involves potential modification of instance knowledge. If any potential modification has to be made, members need to follow the software engineering knowledge defined in the software engineering ontology in order to have a consistent view and same understanding. In the Suggestion Platform, all proposed solutions are initially pending.

After a certain time of gathering some feedback from team members, the decision maker system in the Solution Decision Platform, or Solution Platform for short, later determines what action is needed or which solution will be updated. The actual updating process is also carried out through the Solution Decision platform. A final solution of the discussion is revealed by the Solution Platform. The software engineering ontology gets updated in the ontology repository at the stage.

The issue raised plays an important role in the multi-site distributed software development. Issues are distinguished as either minor or major problems. Access control in the Navigation Platform determines whether an issue is major or minor. Being a major issue, it needs brainstorming through the Suggestion Platform. A minor issue may mean (daily) basic update of instance knowledge by the member in team who has got permission to do so. Figure 1 shows a flow chart of the processes when an issue arises.

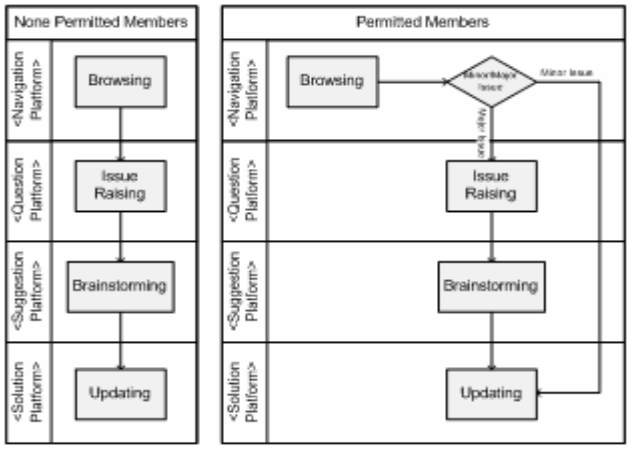

Fig. 1 A flow of the processes when issue arises

Basically, there are two cases. The first case is when an issue is raised by a member who has no access. This is a case of the member not being part of the working team where the issue has arisen. The issue from the member is then considered as a major issue. For example, members in the design team may raise an issue relating to the requirements part which is the responsibility of the analysis team. The issue raised is categorised as a major issue that needs authorisation. An issue raised by a member who has access is another case. This is a case of the member being in the working team where the issue has arisen. In this case, the issue can be considered as either minor or major and this is determined by the member alone. In deciding that the issue is a minor one, it could simply be a matter of updating project data that occurs within the team. The issue, however, can be considered as a major issue if the member decides that the issue is significant enough to need brainstorming from other members or even from other teams.

In particular, platforms improve the process of sharing knowledge so that it is understood in a consistent manner. Also, they facilitate effective and efficient remote communication in the multi-site setting. Details of each platform are given in the next sections.

3. Navigation Platform

Software engineering knowledge, formed into software engineering ontology, helps communications among team members and provides consistent understanding of the domain knowledge. Software engineering ontology, together with its instance knowledge, is used as a communication framework within a project, thereby

providing rational and shared understanding of project matters. In the Navigation Platform, software engineering instance knowledge, in accordance with domain knowledge that is described in software engineering ontology, is extracted. By consulting the software engineering ontology, the platform enables references of software engineering domain knowledge and enables extraction of instance knowledge. For example, class diagrams referred to in the software engineering ontology assert how a set of classes is formed in the diagram.

The specification imposing a structure on the domain of class diagrams i.e. elicitation of each class consists of class name, class attributes, class operations and relationships hold with other classes. Using software engineering domain knowledge, together with instance knowledge, the Navigation Platform dynamically and automatically acts for a certain class instance that the member navigates to retrieve accordingly attribute instances, operation instances, and relationship instances together with the related class instance details.

For example, Class instance CR_Customer is navigated to consequently retrieve ClassAttribute instances and ClassOperation instances. ClassRelationship instances can also be navigated to consequently retrieve Class instances that hold in the relationship and applicable properties of the relationship. In accessing ClassAssociation instance, Class instances held in the relationship and properties like role name and cardinalities are automatically retrieved. Similarly, if those Class instances are accessed, then a list of ClassAttribute instances and a list of ClassOperation instances are retrieved to show its attributes and its operations respectively. In accessing each ClassAttribute instance, details of attribute's name, attribute's data type, and attribute's visibility are shown as referred to ClassAttribute ontology in the software engineering ontology. Navigating ClassAttribute instance CR_CustomerID, its name of 'Customer ID', its data type of 'integer', and its visibility of 'public' can be revealed. The same as ClassOperation ontology referred in the software engineering ontology, in accessing each ClassOperation instance, details of operation's name, operation's visibility, and operation's parameters and parameters' data type can be retrieved.

Moreover, indicating how concepts are inter-related constrains the possible interpretations of terms. For example, with terms class, object and component, sometimes software engineers from different sites, by obtaining the knowledge differently, may easily interpret these terms differently and have difficulty distinguishing between them. The structure of class, object and component respectively imposes differences in the software engineering domain. Therefore, through class diagrams, classes, class attributes, class operations, and/or relationships amidst classes are expected, whereas by declaring object diagrams, objects, object attributes, belonging classes, and/or relationships between objects are expected. By indicating component diagrams, which are apparently different from class diagrams and object diagrams, components, interfaces and/or relationships among components are expected.

Different constituents in the domain of class diagram object diagram, and component diagram in the navigation of class, object, and component respectively reveals divergence in their usage. This is some kind of consensus among the software engineering community or project teams. This then eliminates ambiguous concepts or terms. With team members having the same understanding of concepts, remote communications proceed smoothly and effectively.

4. Question Platform

Both the Question Platform and the Suggestion Platform involve proposals for dealing with multi-site project issues. However, each is devised for different purposes. In the Question Platform, team members propose an issue, whereas, in the Suggestion Platform, they propose the possible solutions for the issue. A member interacts with the platform in order to raise an issue. To do so, the member firstly needs to clarify the problem specifying particular instance knowledge involved which can be fetched from the Navigation Platform. Such issues raised are distinguished as either minor or major issues. A minor issue is simply one in which the change needed does not have to be authorised. This is in the case of instances update within a team such as daily update by authorised members. An authorised approval is needed for a major issue. This is in the case of an instance change by a member in another team. For example, a member in an implementation team has raised the issue of change request on the design part which is the responsibility of the design team. The instance minor changes get updated in the ontology repository directly through the Navigation Platform and are recorded and acknowledged through the Solution Platform. The major instance issues are pending through the Question Platform and make issues outstanding until feedback and suggestions from members are made through the Suggestion Platform.

The three cases of raising major and minor issues are given as follows. The first case is that the member does not have authorisation to make any changes. The member can only view the information and can also raise issues over the information. The second case is one where the

member does have authorisation to make the changes. This case is considered as a major issue due to the member's intention of making the issue outstanding through the Question Platform and getting suggestions from other members through the Suggestion Platform. The third case is where the member does have authorisation to make the changes. The changes made simply involve updating, hence there is no need to go through the Question Platform, instead, the member only needs to go through the solution to record changes and be acknowledged by other members.

In general, all the issues were resolved by members utilising the discipline of the software engineering represented and expressed in the software engineering ontology to gain a common understanding among the teams. This is carried out by the platforms referring to the software engineering ontology. The software engineering knowledge formed in the software engineering ontology is captured widely and thoroughly. For example, in the statechart diagrams ontology, state details, i.e. do, entry, and exit action of UML class and UML class operation or even free text, are captured for anyone who uses one or two of them or all of them.

5. Suggestion Platform

The Suggestion Platform interactions involve modifying instance knowledge. The modifications include add, add new, delete, and edit instance knowledge. However, modifications made in the Suggestion Platform are not instantly updated to the knowledge base in the repository. Rather, the modifications made are pending and will be updated in the Solution Platform if they are applicable.

Mainly, the Suggestion Platform is concerned with a number of proposed possible solutions regarding the problem raised from the Question Platform. This allows all team members the freedom to advocate. In order not to be too much restrict, the software engineering ontology is made open for the worth wide ideas. However, any changes that have been proposed need to be accepted by software engineering domain knowledge asserted in software engineering ontology. For example, in the activity diagrams ontology, the constraint in the fork transition states that exactly one flow of control splits into at least two flows of control. If it is not in line with this notion, it is then not a fork transition. Similarly, join transition restricts joining of at least two incoming transitions and exactly one outgoing transition.

Similarly, in the defect identification ontology, it asserts that there can be non failure or one failure or many failures found however if there is failure(s), there must exist defect(s) or fault(s). This restricts our use of these concepts.

After a certain time of accumulating various ideas from team members, determination of the final solution will be processed. The final solution will then be passed on to the Solution Platform for updating the ontology repository as well as informing participating members.

6. Solution Platform

The Solution Platform is where any decision made is updated. Also the platform informs participated members of decision made. For this study, the basic techniques of voting together with individual reputation value play the roles of determined final solution. However, it can be open to company or organisation policy to draw and develop the making decision system for this platform.

The strategy for updating instance knowledge is given here. For example, Figure 2 (a) shows a use case diagram; Figure 2 (b) shows a revised use case diagram needing to be updated. The use case diagrams used as an example here are derived from the book of Enterprise Java with UML [3].

A list of updating actions is as following.

- Adding new instances ChangePassword and Login for concept UseCase.
- Adding new instance for concept IncludeRelationship relating relations Related_Use_Case with concept UseCase instance ChangePassword and Relating_Use_Case with concept UseCase instance Login.
- Adding new instance for concept AssociationRelationship relating relations Related_Use_Case with concept UseCase instance ChangePassword and Relating_Use_Case with concept Actor instance AdministrativeUser.
- Adding new instance for concept AssociationRelationship relating relations Related_Use_Case with concept UseCase instance Login and Relating_Use_Case with concept Actor instance AdministrativeUser.
- Adding new instance for concept AssociationRelationship relating relations Related_Use_Case with concept UseCase instance ChangePassword and

Relating_Use_Case with concept Actor instance Employee.

- Adding new instance for concept AssociationRelationship relating relations Related_Use_Case with concept UseCase instance Login and Relating_Use_Case with concept Actor instance Employee.
- Adding new instance for concept AssociationRelationship relating relations Related_Use_Case with concept Actor instance Employee and Relating_Use_Case with concept UseCase instance CreateEmployee

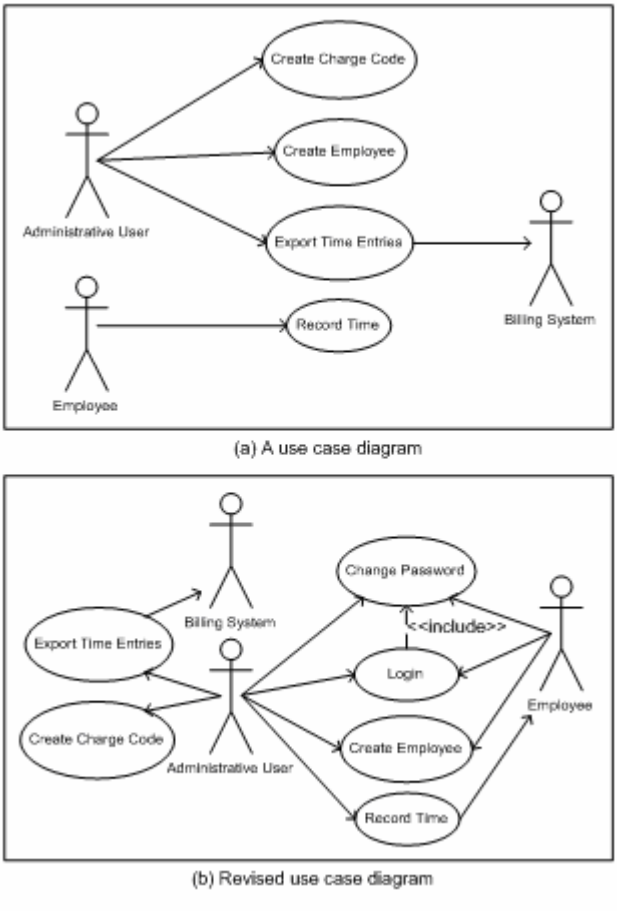

Fig. 2 An example of revising use case diagrams ontology instances

Updating instance knowledge can be as simple as just adding new information on it, or it can be complicated with a mix of deleting and then adding information.

7. Practical Uses of Platforms

Practical uses of the four platforms are given in this section through examples. We start with examples of analysis of problems encountered in the multi-site environment. Then, the use of platforms in resolving these particular problems is explained.

7.1 Problems Analysis

The text transcription is difficult because work is carried out in an environment where development teams are geographically distributed and team members are involved in many projects simultaneously. It is a typical means of global communication which is, however, neither efficient nor sufficient for a multi-site environment. Figure 3 shows such an example of the text transcription that, in a multi-site environment, makes less of an impression.

I am struggling to understand why we need it. I think the system will be simpler for people to understand if we deleted the insurance registered driver.
My reasons for this are that the insurance registered driver is a sub type of the customer. This means that for every insurance registered driver object there must be a corresponding customer object. However, in the customer object we store values like customer type, insurance history value and rental history value. It does not make sense to have these values for the insurance registered driver. I also think people will be confused because we have the rental registered driver as an association with the rental customer (which is a sub type of the customer) but the insurance registered driver is a sub type of the customer.

Fig. 3 An example of text transcription which is not efficient or sufficient for multi-site communication

With ontology-based software engineering, the software engineering terms can be parsed with software engineering ontology concepts and can recall the necessary details and relevant information. We see from figure 13 that it involves the terms of class (class insurance registered driver, class customer, and class rental customer), subclass (sub type), property (property customer type, property insurance history value, and property rental history value), and object (object insurance registered driver and object customer). Terms class, subclass, property, and object apply respectively to the concepts of class, generalisation relationship, class property, and class object in the software engineering ontology. By specifying ontology class instances, relevant information of those instances can be discovered dynamically and automatically.

This is carried out by referring to software engineering ontology which asserts that concept class has its semantic of containing attributes, operations, and relationships holding among other classes. Automatically drawing out details facilitates others' greater understanding of the content, thereby reducing misunderstanding, and eliminating ambiguity.

In the case of team members working at the same time on several projects, the contents of that work are similar. For example, one project is about car rental systems and the other project is about car dealership systems both of which have details of car and customer. This easily causes confusion with the term 'car' and distinguishing between car rental systems and car dealership systems. With ontology-based software engineering, the instance knowledge, being an instance of ontology classes, does use unique name assumption. The unique name assumption, which is like a primary key (one name for each instance), helps prevent ambiguity.

7.2 Platforms Uses

Examples of the practical uses of platforms are given throughout this section. We start with the case of an issue being raised by a member who has no access. For example, an issue may relate to a project design raised by a programmer in the implementation team. This is considered as a major issue. The Navigation Platform is where the software engineer first identifies the involved data. The user raises the issue by fetching relevant instance knowledge from the Navigation Platform to the Question Platform. Figure 4 shows the class diagram that the member raised the issue on.

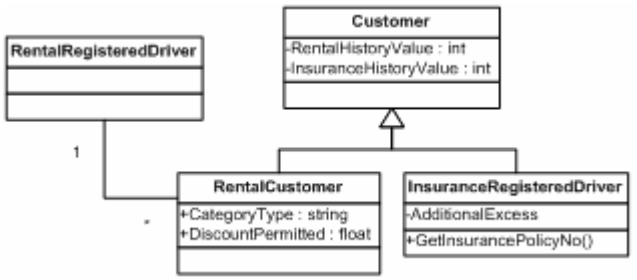

Fig. 4 Issue on an UML class diagram

The Question Platform is where members raise problems they consider to be issues. Everyone can see the issue being raised. This allows brainstorming over the issue and possible solution proposals can then be made in the Suggestion Platform.

From here, there are two possible solutions proposed in the Suggestion Platform. Figure 5 shows the first solution proposal and figure 6 shows the second solution proposal. As stated earlier, either of these two potential solutions is not yet a solution. Throughout the platform, worthwhile suggestions can be carried out by: deleting, adding, editing, or by a mix of, for example, deleting and then adding the instances.

As from figure 5, the suggestion is to delete concept Class instance InsuranceRegisteredDriver. Another suggestion from Figure 6 is to first delete concept ClassAttribute instances RentalHistoryValue and InsuranceHistoryValue relating relation has_Attribute with concept Class instance Customer. Secondly, new instance RentalHistoryValue is added for concept ClassAttribute relating relation has_Attribute with concept Class instance RentalCustomer. Thirdly, a new instance InsuranceCustomer is added for concept Class. Fourthly, new instances InsuranceCustomerID and InsuranceHistoryValue are added for concept ClassAttribute relating relation has_Attribute with concept Class instance InsuranceCustomer. Fifthly, a new instance is added for concept ClassGeneralisation relating relation Related_Object_Class_Component with concept Class instance InsuranceCustomer and relating relation Relating_Object_Class_Component with concept Class instance Customer. Lastly, a new instance is added for concept ClassAssociaton relating relation Related_Object_Class_Component with concept Class instance InsuranceRegisteredDriver and relating relation Relating_Object_Class_Component with concept Class instance InsuranceCustomer.

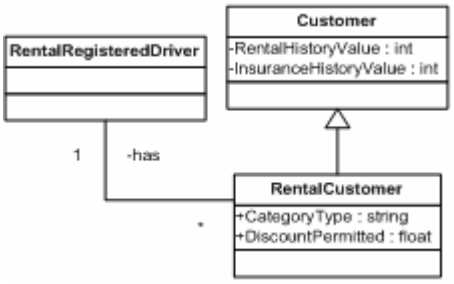

Fig. 5 The first solution proposal

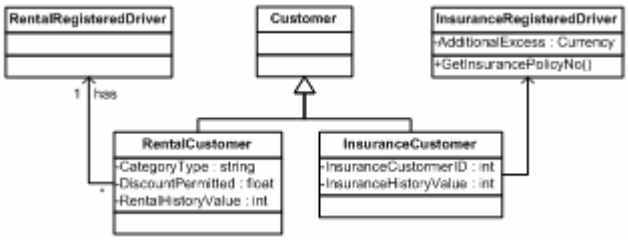

Fig. 6 The second solution proposal

The proposed changes become solution changes in the Solution Platform. Until such decision maker systems in the Solution Platform determine the final solution, instance knowledge gets updated along the lines of the chosen solution.

The following examples show the case of an issue raised by a member who has access. In this case, it is the member alone who determines whether the issue is major or minor. In the case of a minor issue identified by the member, it is simply a matter of updating project data through the

Navigation Platform, and the Solution Platform retains and displays the record. This is no need to go through question and Suggestion Platforms because its purpose is to update in order to share instance knowledge.

Figure 7 (a) shows an original activity diagram, while Figure 7 (b) shows an updated activity diagram. The activity diagram used as example here is derived from the book of Enterprise Java with UML [3].

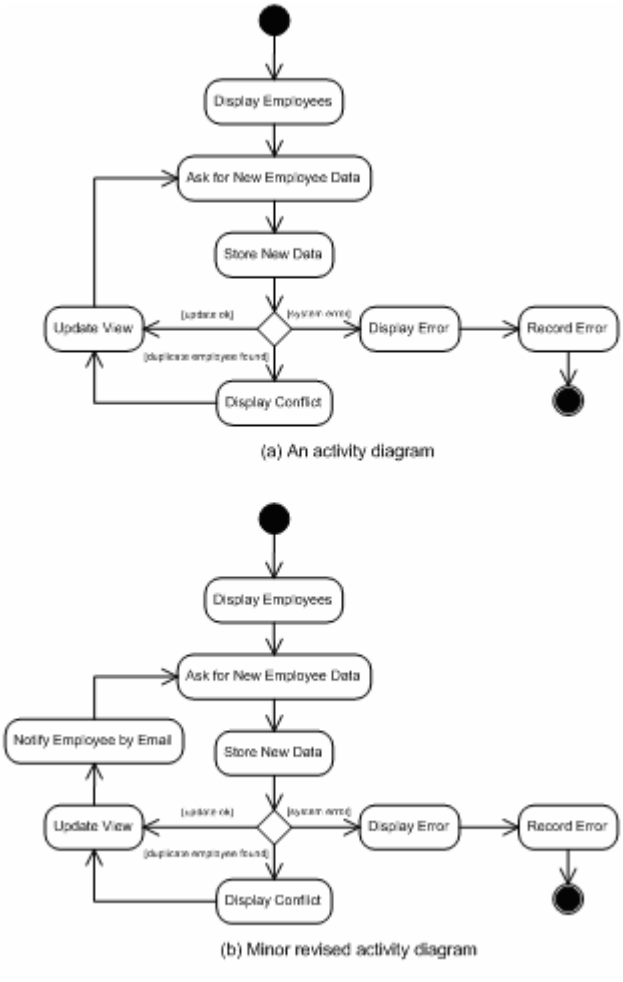

Fig. 7 Minor issue of the activity diagram revision

As can be noted when comparing figure 7 (a) and figure 7 (b), the software engineer has revised the transition of activity 'Update View'. Originally, activity 'Update View' transited to activity 'Ask for New Employee Data'. Revision has been made by activity 'Update View' transited to activity 'Notify Employee by Email' and activity 'Notify Employee by Email' transited to activity 'Ask for New Employee Data'. Functioning is as following:

- Delete concept NormalTransition instance that has relation Related_Activity with concept Activity instance named 'Ask for New Employee Data' and has relation Relating_Activity with concept Activity instance named 'Update View'.
- Add new concept Activity instance named 'Notify Employee by Email'.
- Add concept NormalTransition instance that links relation Related_Activity with concept Activity instance named 'Notify Employee by Email' and links relation Relating_Activity with concept Activity instance named 'Update View'.
- Add concept NormalTransition instance that links relation Related_Activity with concept Activity instance named 'Ask for New Employee Data' and links relation Relating_Activity with concept Activity instance named 'Notify Employee by Email'.

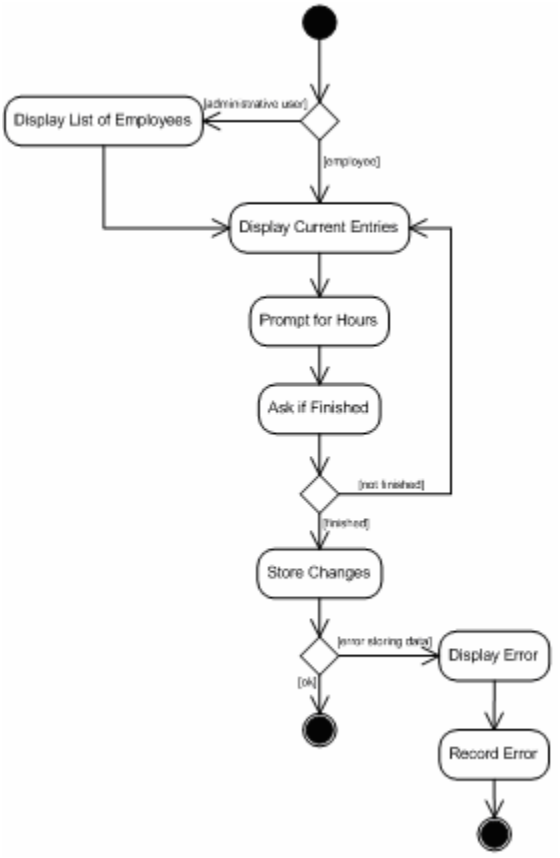

Fig. 8 An activity diagram

In the Navigation Platform, an authorised member can directly update project data using add, delete, update, add new functions or else despatch this as a major issue using the fetch function to post to the Question Platform. In the case where a member considers an issue to be a major issue, the process involves going through navigation,

question, suggestion, and Solution Platforms. For example, figure 8 shows an activity diagram which is the point at issue and figure 9 shows an activity diagram that is ultimately suggested to resolve the issue. The activity diagram used as example here is derived from the book of Enterprise Java with UML [3].

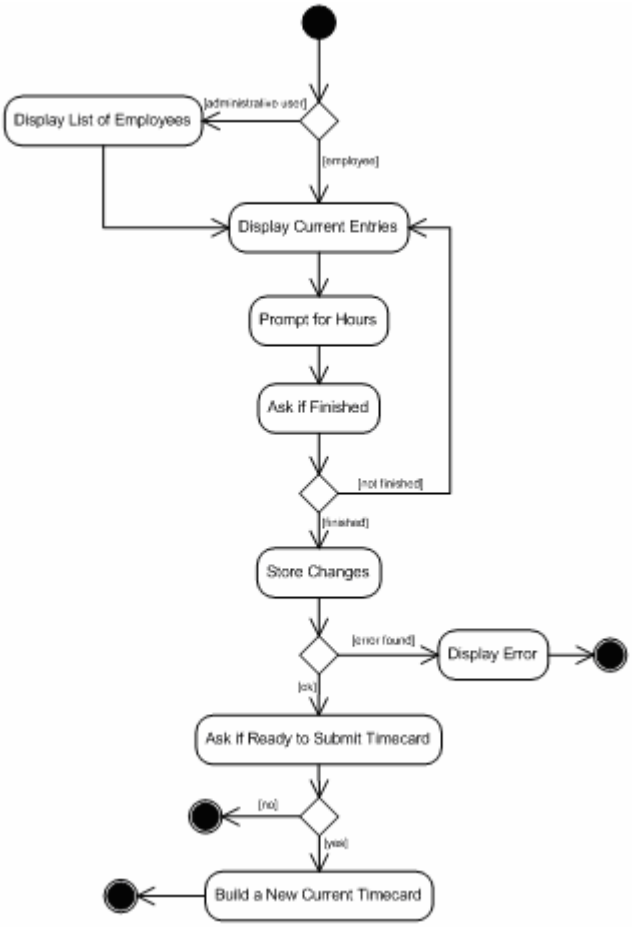

Fig. 9 Major revised activity diagram

A member starts progressing towards a resolution of the issue by fetching involved instances from Navigation Platform to Question Platform. Consequently, in Suggestion Platform a dummy solution is suggested. After brainstorming, the solution is given in the Solution Platform and the ontology gets updated. Along the line of Solution Platform in this example, a list of functions is as follows:

- Update BranchTransition instance linked to Related_Branch_Activity_1 instance named 'Display Error' which the Guard_Expression_1 instance is changed to 'error found'.
- Add new Activity instance named 'Ask if Ready to Submit Timecard'.
- Update BranchTransition instance linked to Related_Branch_Activity_2 instance which is Stop instance. Delete the Stop instance and add the Activity instance named 'Ask if Ready to Submit Timecard'.
- Add new Activity instance named 'Build a New Current Timecard'.
- Add new BranchTransition instance which links Relating_Branch_Activity_1 instance to Activity instance named 'Ask if Ready to Submit Timecard', links Related_Branch_Activity_1 instance to Activity instance named 'Build a New Current Timecard', and links Related_Branch_Activity_2 instance to Stop instance. Its Guard_Expression_1 is a string of 'yes' and its Guard_Expression_2 is a string of 'no'.
- Add Stop instance that links Relating_Special_Activity named 'Build a New Current Timecard'.
- Add another Stop instance that links Relating_Special_Activity named 'Display Error'.

8. Discussion

Based on the functioning of the four platforms, observations are summarised in this section. We compare the four platforms functioning with multi-site issues of communications, coordination, cooperation, awareness, interoperability, track and trace, and just-in-time. This is to illustrate that these issues can be overcome. We divided our discussion into three main sections. They are:

- Multi-site Communications
- Remote Management
- Multi-site Project Tools

8.1 Multi-site Communications

The platforms pose as a communication framework for multi-site distributed software development. This overcomes multi-site issue of communications.

Communications

The disadvantages associated with multi-site communications rather than face-to-face communications in a software development, whose tasks are distributed across remote sites, is a key issue. Platforms are a means

of exchanging information with explicit details for the collaboration between multi-site teams. Basically, the platforms being a communication framework are the place to find answers in a multi-site distributed environment. The Navigation Platform is available at any time for remote team members to seek further more information, to clarify a mutual agreement, and to share information. The Question Platform is where discussions are conducted on multi-site project issues. The Suggestion Platform is used for brainstorming purposes and to discuss issues that have been raised. The Solution Platform assists the project to keep moving forward and here also the project data gets up-to-dated.

The underlying architecture of the communication framework is the software engineering ontology which identifies the domain knowledge of software development along with instance knowledge of project data. Knowledge sharing through the software engineering ontology eliminates misunderstandings, miscommunications, and misinterpretations. Software engineering ontology presents explicit assumptions concerning the objects referring to the domain knowledge of software development. A set of objects and interrelations and their constraints renders their agreed meanings and properties. For example, the confusing terms of 'classes, 'objects', and 'components' in object-oriented software development can be simplified, and when perceiving them, software engineers agree to recognise their constitutes, their interrelations, and their constraints.

8.2 Remote Management

Platforms make things easy to manage, even remotely. Multi-site issues of coordination, co-operation, awareness, and interoperability have been overcome.

Coordination

Remote team members' coordination is important, especially in multi-site software development. The Navigation Platform assists team members to be aware of tasks to be, or being, carried out. Sharing their project information through the Navigation Platform makes project tasks explicit; hence, this makes it easy for others to coordinate the work and be aware of it. For example, if there is an ignorance of tasks, then there can be no sharing of information. Wrong tasks that have been carried out are identified. In this case, the Suggestion Platform is where members can coordinate meetings to discuss task misinterpretation. Therefore, the potential problems, such as two groups overlapping in some work, or other work not being performed due to misinterpretation of the task, are no longer the case.

Cooperation

Instance knowledge is drawn based on a consensus of domain knowledge of software engineering formed in the software engineering ontology underlying the platforms. Hence, commonalities relating to the knowledge are assumed. This specially overcomes different knowledge of team members as coming from diverse backgrounds. Additionally, failing to describe their local context when team members raise project issues or share information and knowledge will not be the case no more. This is because team members are referring the consensus software engineering domain knowledge formed in the software engineering ontology.

Awareness

The process of going through navigation, question, suggestion, and Solution Platforms is undertaken when there are issues that need authorising and brainstorming from team members in multi-site projects. This produces awareness of issues that have been currently raised (from the Question Platform), the issues that have been clarified (from the Solution Platforms), any task that has been misunderstood (from the Solution Platforms), and an understanding of the reason(s) for a team member not following the project plan (from the Suggestion Platform). Throughout, instance knowledge explicitly displayed on the platforms specifies the current status of the project particularly from the Navigation Platform. Team members are made aware of work that has been done, work being done as planned, work being done to incorporate other tasks, and the team members who are performing the tasks.

Interoperability

Software engineering domain knowledge is captured in the form of software engineering ontology. Software engineering standards, rules, and formalisms facilitate the production of high-quality task performance. Hence, the development of components from different sources and their ability to integrate these components to build a system can be relied on. If component integration has not been successful at some point during integration, the issue is raised through the Question Platform in the line before the project fails. Additionally, mutual adjustment, feedback and discussion during integration, in order to align team members' actions, can be carried out through the platforms.

12.3 Multi-site Project Tools

Platforms define how tools are utilised. This enables one to overcome the issues of track and trace and just-in-time in the multi-site situation.

Track and trace

The platforms are where members keep track of project data being exchanged, discussed, or shared and also member relationships. Local context and constraints and issues do not remain local, but rather are shared to enable brainstorming across multi- sites. In particular, the Solution Platform is where members are advised of any updated project data trace. The platforms are available online for remote teams to send and respond to the information at any point in time.

Just-in-time

The software engineering ontology underlying the platforms provides a source of precise and explicit software development terms or concepts which can be communicated across remote team members, multi-site projects, and applications including intelligent agents. This man-machine system, through the platforms, makes it possible for members to perceive their counterparts as they are sharing the same knowledge formed in the software engineering ontology. Additionally, automatically and immediately, members send and receive responses to sharing project data formed as instance knowledge.

9. Conclusion

Detailed specific management systems of safeguard, ontology, and decision maker have been given. We have presented the four platforms i.e. Navigation Platform, Question Platform, Suggestion Platform, and Solution Platform for multi-site distributed software development and knowledge sharing. Through examples, we have also presented the practical uses of platforms. We have discussed the platforms' benefits.

References

[1] Wongthongtham, P., et al., Ontology-based multi-site software development methodology and tools. Journal of Systems Architecture, 2006. 52(11): p. 640 - 53.
[2] Wongthongtham, P., A methodology for multi-site distributed software development, in School of Information Systems. 2006, Curtin University of Technology: Perth.
[3] Arrington, C., Enterprise Java with UML. 2001, New York, USA: John Wiley & Sons, Inc.

Dr. Pornpit Wongthongtham received the M.Sc. in Computer Science from Chulalongkorn University, Thailand, and PhD degrees in Information Systems from Curtin University of Technology, Australia, in 1999 and 2006, respectively. Currently, she is working as a research fellow at the Digital Ecosystems and Business Intelligence Institute (DEBII), Curtin University of Technology