Business Process Ontology for Defining User Story

Chaleerat Thamrongchote
Department of Computer Engineering
Faculty of Engineering, Chulalongkorn University
Bangkok, Thailand
chaleerat.t@student.chula.ac.th

Wiwat Vatanawood
Department of Computer Engineering
Faculty of Engineering, Chulalongkorn University
Bangkok, Thailand
wiwat@chula.ac.th

***Abstract*—Writing a user story is essential to communicate with end users and developers. It is popular practices found in most agile methodologies. Therefore, to reuse the user stories from the previous successful projects becomes necessary. In this paper, we propose the business process ontology for defining user story. The ontology is a knowledge base designed to collect the user stories in term of N-triple. The ontology schema is designed using classes and hierarchy relation along with the properties according to Role-Action-Object relations. We introduce the synonym property to simplify and reduce the number of the nodes in the ontology as well. Finally, we demonstrate how to initially populate and build the ontology from the historical user stories.**

Keywords— Ontology; Business Process; User Story;

I. INTRODUCTION

Writing user story is one of the most popular techniques in agile methodology [1] such as extreme programming (XP), scrum, crystal family of methodologies and feature-driven development (FDD) [2]. The user stories describe customer needs on the functionality of software product. User stories must be easy to understand, short sentence and concise clarity. They help customer and developer with a consistent understanding on the operation of the software product. Therefore, there are several starter guidelines to write the good user stories which specify who does what and why to do it. Although some templates are easy to use, how to choose words to write on user story is still difficult without guidance on domain specification. In most cases, the words defined in the user stories are referred back to the historical project data.

In this paper, we propose an alternative mean to write user stories more easily. Using the collected knowledge concepts of the related or similar business processes which have been done successfully. For example, when we assign who perform activity then the rest of the possible stories would be reasonably suggested according to the previous successful projects.

This paper is organized as follows. The introduction is described in section I. Section II reviews the backgrounds and the related works. Section III describes how business process ontology for user story work and section IV shows the case study and results. The conclusion is shown in section V.

II. BACKGROUNDS AND THE RELATED WORKS

A. *Ontology*

Ontology is a formal model to describe the interesting domain concepts. It serves as a knowledge base of the terms and their relationships [1]. Ontology knowledge base contains the following elements [2] - *Classes* are the scope of knowledge or a particular subject and can explain the details, *Properties* are the features which are used to elaborate ideas, *Relations* are the ways in which classes are related among them, *Role restrictions* are the limitations of concepts being defined, *Individuals* are the things that ontology describes such as people, animal, automobiles and machines, as well as abstract individuals such as numbers and words. Ontologies describe features or attributes of classes by slot or properties. In some classes, it may be subclass to describe it.

B. *User stories Template*

User stories are short sentences written on small cards telling the requirements and functions a user needs. Each user story on a card would tell only one function only. The most common user story format is described as follows [6]:

As a <type of user> I want <some goal> so that <some reason>.

Each part of the above user story format is defined as follows - *<Type of user>* represents user of the software system, which is either a role or another requesting system, *<some goal>* represents user's purpose or intention expected from the software system, *<some reason>* represents user's reason indicating why to achieve this user story. Moreover, in order to be more specific, the user story template for object-oriented application [7] provides explicit action and object of the user stories [8]. This template is shown in Fig. 1.

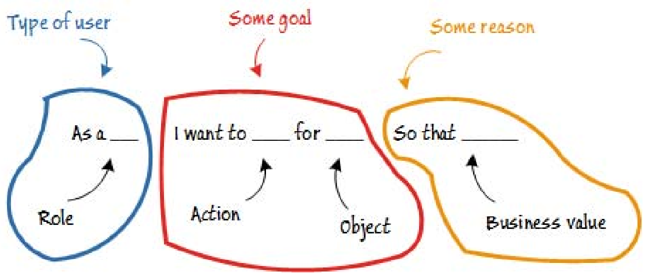

Fig. 1. Object-oriented Specific User Story Template [7]

Each part of the object-oriented specific user story template [7] is defined as follows - *Role* represents who can operate activity or who may benefit from it including the interacting system which triggers the software system, *Action* represents user's action in the activity, *Object* represents the entity which is done by the given action, *Business value* represents the expected benefit from this story and reason for action.

We consider the object-oriented specific user story template in our approach so that it can parse the part of a goal to make it easier to write a user story because of the most common user story format is quite difficult to control the form of a sentence. Besides, we could map it correspondingly into each N-triple in our ontology schema. By using the Role, Action and Object parts, we finally assign the Role-Action-Object relations. Our initial business process ontology would be populated using the project historical user stories.

III. Our Business Process Ontology Design Scheme

In this section, we demonstrate how to initially build our business process ontology. In Fig. 2, we describe our design scheme to define the relevant ontology schema graph and how to populate the individuals into the resulting ontology. In order to optimize our business process ontology, we apply the concept of the synonym for words to our initial raw ontology so that the number of individuals significantly decrease. In our design scheme, the user stories written in the object-oriented specific template [7] are collected and analyzed. We firstly consider and build the hierarchy of the words found in each part of the user stories. Then, we assign the relationship between words using our defined properties according to the Role-Action-Object relations found in the user story template. Finally, we optimize these initial raw ontology graph using node collapse and synonym concepts.

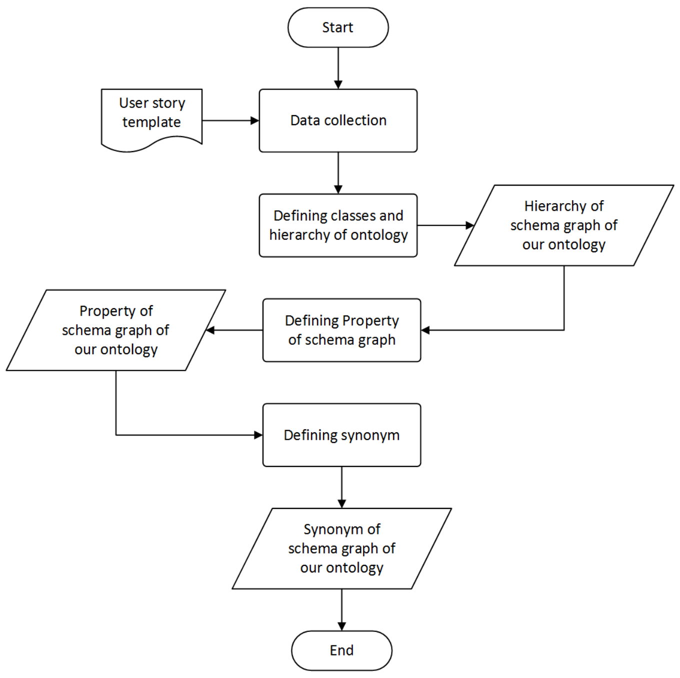
Start
User story template
Data collection
Defining classes and hierarchy of ontology
Hierarchy of schema graph of our ontology
Property of schema graph of our ontology
Defining Property of schema graph
Defining synonym
Synonym of schema graph of our ontology
End

Fig. 2. Our Business Process Ontology Design Scheme

A. Data collection

The user stories are collected from the historical project data to be analyzed. The user stories are written using the object-oriented specific user story [7] shown in Fig. 1. We focus on three sets of words extracted from Role, Action, and Object part of the user stories.

B. Defining classes and hierarchy of ontology

Given three sets of words from the previous step, the words of Role, Action, and Object are about to be added with the classes and their hierarchy which is the fundamental knowledge concept in the ontology. The hierarchy of the words is defined as a tree with a root called *RootWord* and its children node called *SpecificWord*, shown in Fig. 3. The *RootWord* is the most generic words found in the business domain and a *SpecificWord* is supposed to be a specific type of its parent. We also call parent node as superclass and the child node as a subclass. The hierarchy of words is defined as a tree with the depth of N level so that any *SpecificWord* could have its own children or subclasses. A dotted arrow from subclass to its superclass represents the subclass relation is also shown in the tree.

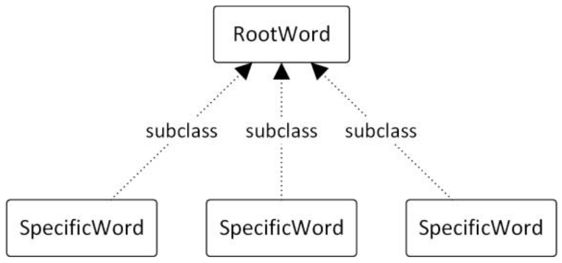

Fig. 3. Hierarchy of schema graph of our ontology

C. Defining Property of schema graph

Given a set of individuals assigned with their subclass relationship from the previous step. We are about to assign additional property lines to indicate the Role-Action-Object relations found in the user story statements. We define two new properties called *[performAction]* and *[performObject]* in our ontology schema graph. The [performAction] property links from Role class to Action class while the *[performObject]* links from Action class to Object class. For example, "*SpecificRole [performAction] SpecificAction*" means that the specific role individual performs the specific action individual. Then, "*SpecificAction [performObject] RootObject*" means the specific action individual use or perform on the root of object individual. The root object individual practically refers to all of its specific individuals. As shown in Fig. 4, each role individual carries a property link to a particular action individual which carries another property link to an objecting individual, in order to represent one single user story.

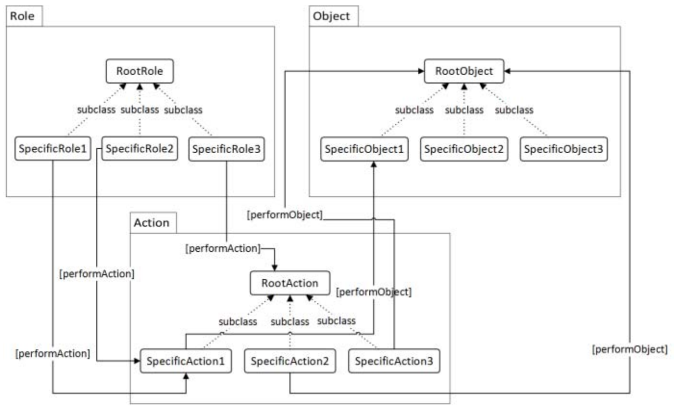

Fig. 4. Property of schema graph of our ontology

We now accomplish the initial business process ontology built from the collection of user stories. This ontology is capable of guiding us to reuse the relevant story when needed. Starting with a selected role individual, only the relevant set of the action individuals, linked by our ontology, are the valid choices for the story. It is semantically effective guideline scheme rather than choosing from the whole available action individuals. Whenever needed, a new user story may be considered and the new role, action, and object individuals could be merged into the existing ontology graph.

D. Defining synonym

The initial version of the business process ontology from the previous step contains too many nodes of individuals and relationships. We apply the concept of synonym as to simplify the ontology. A property called *[isSynnonym]* is assigned to define the synonym class of the existing one. For example, "*SpecificRole2 [isSynonym] SpecificRole1*" means the specific role2 individual is a synonym of the existing specific role1 individual. The synonym class is totally treated as its original class. The duplicated nodes of individual and their relationship lines are reduced. As shown in Fig. 5, three of *[isSynonym]* property lines are depicted at circle A, B, C. Thus, "*SpecificRole2 [performAction] SpecificAction1*" and "*SpecificAction2 [performObject] SpecificObject3*" are implicitly valid without any direct property links.

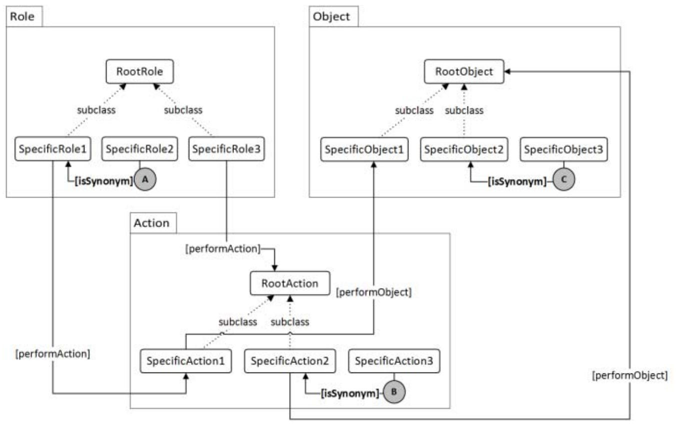

Fig. 5. Synonym of schema graph of our ontology

IV. CASE STUDY

In this section, we propose a case study from the online shopping application in the transaction as a customer wants to shop online in the scenario of view of the product. However, we cannot be available to every class that exists. For this reason, our ontology schema graph just has some part to make it easier to understand because there are a lot of classes that occur in this case study. If we show all the possibilities, it may be a confusion. The complete ontology schema graph is shown in Fig. 9 (at the end of paper). In addition, we demonstrate how to build business process ontology. In this case study, the user stories tell the functions concerning the payments of the online shopping orders. The customer is classified into member or guest. A guest is allowed only to pay the shopping order by cash while a member has another option to pay by points. Both guest and member customer are allowed to order all specific types of products as shown in Fig. 6.

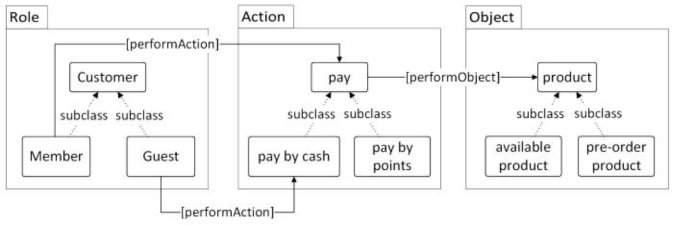

Fig. 6. Case study of schema graph of our ontology

Here are the samples of the valid N-triple of individuals and their relations: Member *[performAction]* pay, pay *[performObject]* product, Guest *[performAction]* pay by cash. Thus, with these three N-triples in our ontology, the following six user stories would be covered:

As a *Member*, I want to *pay by cash* for *available product*, so that I can order product online.

As a *Member*, I want to *pay by points* for *available product*, so that I can order product online.

As a *Member*, I want to *pay by cash* for *pre-order product*, so that I can order product online.

As a *Member*, I want to *pay by points* for *pre-order product*, so that I can order product online.

As a *Guest*, I want to *pay by cash* for *available product*, so that I can order product online.

As a *Guest*, I want to *pay by cash* for *pre-order product*, so that I can order product online.

When it comes to having a totally new class, called *Visitor* as shown in Fig. 7, it would be considered to act as either subclass of the existing class or a new generic class. In this case, the *Visitor* class is assigned as a subclass of the existing *Customer* class. If the new user stories regarding the *Visitor* shows the same stories as the *Guest* class, then the *[isSynonym]* property link would be assigned between the *Guest* and *Visitor* class as shown in Fig. 8.

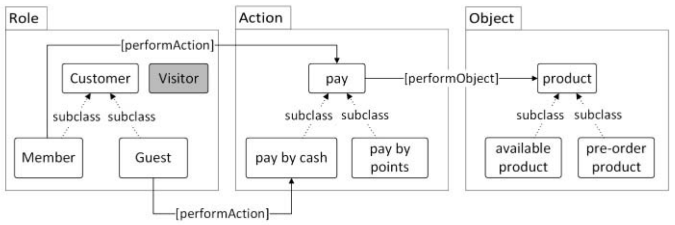

Fig. 7. Case study of adding class of schema graph of our ontology

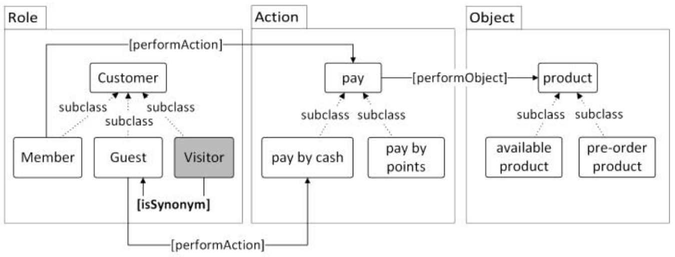

Fig. 8. Case study of synonym of schema graph of our ontology

V. Conclusion

This paper proposes the business process ontology design scheme and demonstrates how the ontology is initially built. The business process ontology is designed as a knowledge base to collect the user stories which are the requirements and functions needed for the software system. It is expected to effectively guide to reuse the archive of the previous successful project to write the user stories in the current project. We consider the object-oriented specific user story template in [7] as our standard to capture the Role-Action-Object relations defined in the user stories. Then, we demonstrate the steps to populate the individuals and their relations in the ontology. The subclass and hierarchy of the ontology are defined along with these properties - *[performAction]* and *[performObject]*. We also introduce the concept of a synonym to simplify and significantly reduce the number of nodes in the ontology.

References

[1] Leffingwell, D., "Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise". Addison-Wesley, 2011.

[2] Abrahamson P, Salo O, Ronkainen J, Warsta J, 2002. Agile software development methods: Review and analysis (Technical report). VTT.

[3] Gruber, Thomas R. A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 1993.

[4] Noy, Natalya F. and McGuinness, Deborah L. Ontology development 101: A guide to creating your first ontology. Stanford University, 2001.

[5] Phillip Lord (2010) Components of an Ontology. Ontogenesis. http://ontogenesis.knowledgeblog.org/514

[6] M. Cohn, User stories applied: for Agile software development. Addison-Wesley, 2004.

[7] A. Zeaaraoui, Z. Bougroun, M.G. Belkasmi and T. Bouchentouf, "User stories template for object-oriented applications," Innovative Computing Technology (INTECH), 2013 Third International Conference on, London, 2013, pp. 407-410.

[8] A. Zeaaraoui, Z. Bougroun, M. G. Belkasmi and T. Bouchentouf, "Object-oriented analysis and design approach for requirements engineering," Innovative Computing Technology (INTECH), 2012 Second International Conference on, Casablanca, 2012, pp. 133-137.

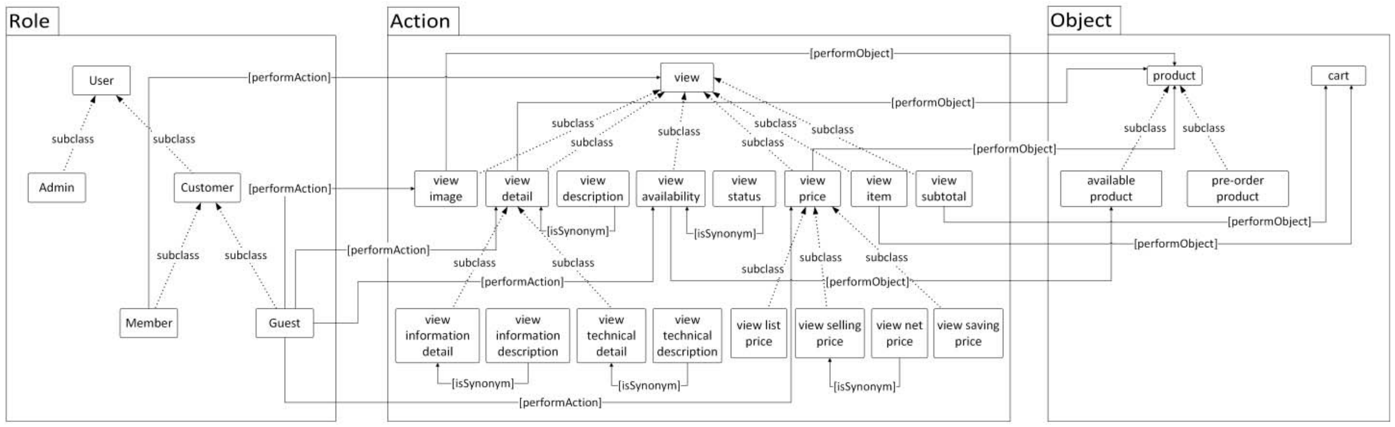

Fig. 9. Scenario of view of product in schema graph of our ontology