Modelling Requirements for Content Recommendation Systems

Sarah Bouraga[1,2], Ivan Jureta[1,2,3], and Stéphane Faulkner[1,2]

[1] Department of Business Administration, University of Namur
[2] PReCISE Research Center, University of Namur
[3] Fonds de la Recherche Scientifique – FNRS, Brussels
{sarah.bouraga, ivan.jureta, stephane.faulkner}@unamur.be

Abstract. This paper addresses the modelling of requirements for a content Recommendation System (RS) for Online Social Networks (OSNs). We model an essential dynamic on OSN, namely that when a user creates (posts) content, other users can ignore that content, or themselves start generating new content in reply. This dynamic is key to designing OSNs, because it influences how active users are, and how attractive the OSN is for existing, and to new users. We apply a well-known Goal Oriented RE (GORE) technique, namely i-star (i*), and show that this language fails to capture this dynamic, and thus cannot be used alone to model the problem domain. We suggest using an additional layer of modelling on top of an i* model, using Petri nets, and discuss the benefits of doing so.

1 Introduction

A particular trait of Online Social Networks is that behavior of one user has an impact on the behavior of other users and of the system itself. When a user shares what we call an event type (an event type is an activity generated by a user that can produce a notification to this users friends, for instance "Share a photo"), the users friends - if they see it - have a choice: they can decide to reply to that event type or not. This decision has an impact on the information that is exchanged on the system. We can also observe that the amount and the order in which the event types are notified to the users vary depending on the OSNs. Some OSNs, such as Tumblr or Twitter, notify each user will all the event types generated by the user's friends. Others, such as Facebook, seem to apply some sort of recommendation technique in order to decide which, and in what order, event types are displayed to the user. This dynamic and these specificities make the OSNs a particular class of systems; the requirements of which may or may not be modeled using existing requirements modelling languages.

On OSNs, a user switches roles constantly between content generator and content receiver. The goals and softgoals are different when the user is generating a post, as opposed as replying to a post. In other words, the user is generating instances of different entities, depending on the role she has: a generator generates instances of a "post", while the receiver generates instances of a "reply".

Therefore, we believe that a RS, which needs to do content recommendation, needs to see these roles as separate.

We consider two OSN users, User A and User B. User A and User B are "friends" on the OSN. A shares something on the OSN, that is, she generates an event type. The OSN has to decide if the event type should be notified to B. If it is, then B has to decide whether to reply to the event type. For instance, if A shares a photo on the OSN, and if the photo is notified to B, then the latter has to decide whether she will like, or comment the photo, or else. If B decides to reply to the event type, then her reply amounts to an event type, and she now acts as generator, that is, if she replies, then User B has generated an event to which other users may choose to reply. Hence, the mechanism goes on. The first round of this dynamic is represented in Figure 1.

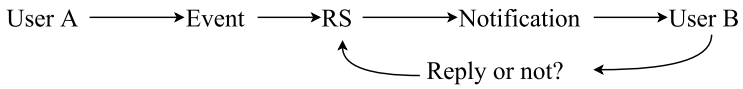

Fig. 1. First Round of Dynamics

2 Research Question and Methodology

This paper addresses a specific case of a general problem, namely the representation and management of requirements that involve some dynamic. The research question is: How can we represent the requirements for RS in one single i* diagram? The reason we want it in one i* model, is that how this dynamic is designed into the OSN influences if the OSN will satisfy some of the basic user goals (such as, enjoy the OSN), and will influence the extent to which important softgoals are satisficed. We want to have one model where we can both do the analysis of the recommendation mechanism, and represent how it influences user goals and softgoals. This question leads to another question: What new concepts and/or relations do we need to use together with those of i* to show the dynamics represented in Figure 1?

In order to address these questions, we apply the following methodology. Firstly, we construct the base layer using i*. It represents what happens on an OSN, but from a static point of view. Secondly, we construct the second layer representing the dynamic aspects of OSN, using Petri Nets. We build this layer by analyzing and identifying what happens when a user shares a post on the OSN. Finally, we connect both layers by lifting up i* symbols to the Petri Net layer.

Our approach is not to extend i* with new concepts, that is, we are not changing the meta-model of i*. Indeed, we do not want our approach to require any new learning. However, we aim to represent the dynamic in one layer, have

the i* model in the other layer, and then have relationships, which connect the two layers.

The present paper is a synthetic version of a longer discussion, which is available and archived online [1].

3 Contribution: The Layers and The Connection Between Them

The first layer is displayed in Figure 2 (where the letters are connectors), and the second is displayed in Figure 3.

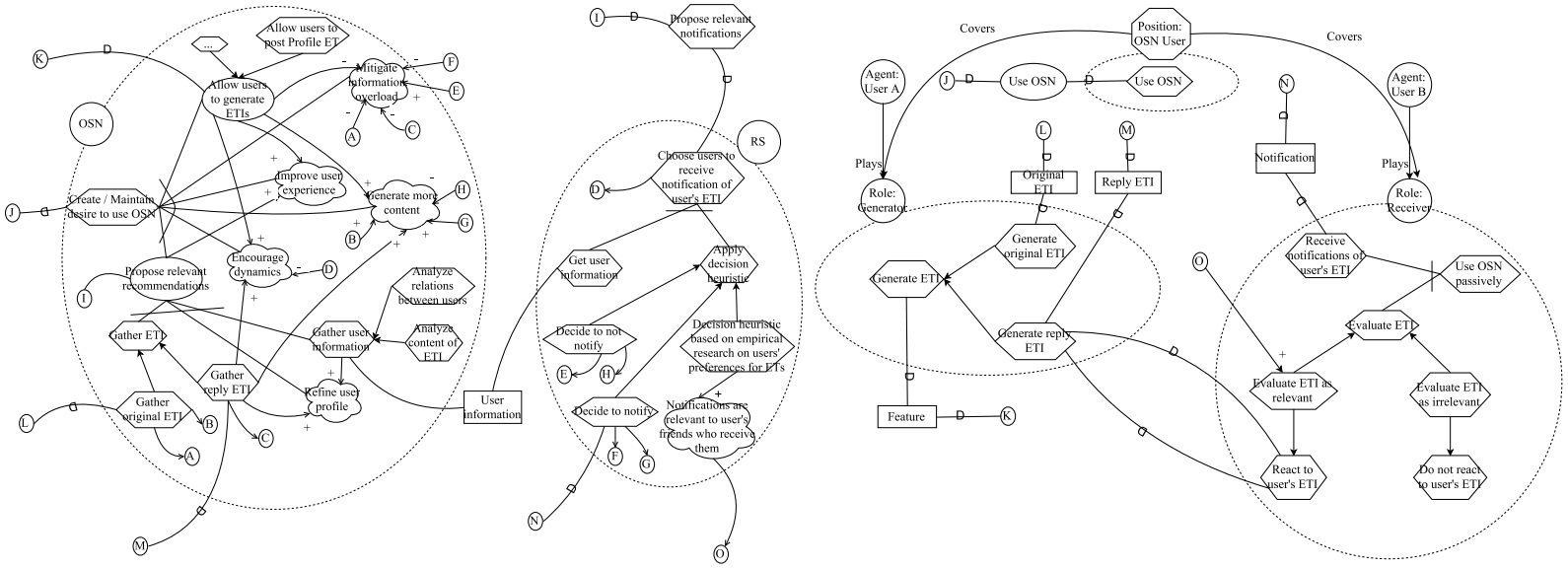

Fig. 2. I* Layer: Strategic Rationale Model

A Petri net is a particular kind of directed graph, together with an initial state called the initial marking, M_0 [2]. Petri net consists of two kinds of nodes: (i) places, and (ii) transitions. Graphically, k black dots (tokens) are represented in place p. A marking is designated by M, an m-vector, where m is the total number of places. The pth component of M, indicated by $M(p)$ is the number of tokens in place p. The Petri Net is built by analyzing the i* layer and by identifying what happens when the generator decides to post an event type. In the Petri Net, we only insert elements that are present in the base layer. With this second layer, the RS has a clear procedure to follow in case the user posts an event type and in case it decides to notify it. Furthermore, a clear distinction is made between User A as generator and User A as receiver (same goes for User

B); which is important for the RS, because depending on who generated the ET (and who will receive the ET), the decision of notify or not notify might be different.

How do these layers connect? The base layer represents the various elements that can occur in an OSN. The second layer represents the dynamic found in the content recommendation context of an OSN and is triggered by the sharing of an original event type. Graphically, the connection occurs as follows. Once the trigger happens, the symbols of the base layer lift up to the second layer. We replace the circles of the Petri Nets with the corresponding symbol of i*. Hence, the model reads more easily; because we directly see to what symbol the circles of the Petri Net correspond. Nevertheless, we do not insert new symbols or new concepts. All the symbols and concepts are known and belong to the i* or Petri Net languages. We just use the Petri Net formalism to sequence the i* symbols.

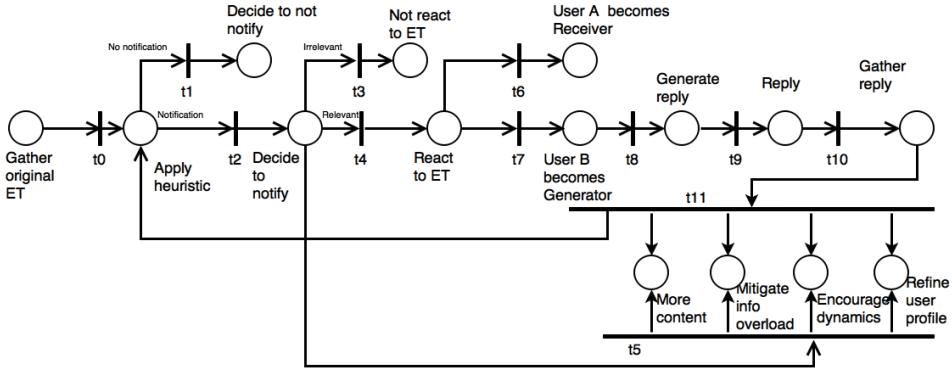

Fig. 3. Petri Net Layer

The "new" Petri Net model is represented in Figure 4. The first layer is thus composed of Figure 2; while the second layer consists of Figure 4. This swap allows a clear connection between the two layers. The relevant content for the modelling of the dynamics is elevated from Figure 2 to Figure 3; where we use the Petri Net procedure to model the observed dynamics.

4 Discussion

The motivating problem of this paper was the modelling of requirements for content recommendation on OSNs. More specifically, we aimed at modelling the mechanism represented in Figure 1. We noticed that the original i* did not allow us to model the dynamics observed on OSNs. We also know that Petri Nets are a nice way to simulate the dynamic behavior of a system [2]. We combined these two standards, using a layer mechanism to model, in one diagram, the requirements of a content RS. The benefits of our approach are threefold.

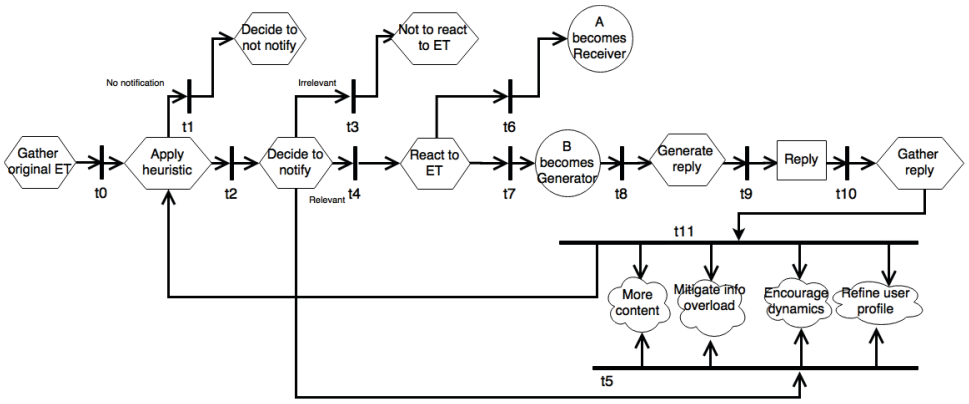

Fig. 4. Connection Between Layers

Firstly, we do not introduce another extension, any new concepts, to an existing language. Hence, the use of our proposal does not require any new learning.

Secondly, the layer mechanism allows us to manage the complexity. Each layer consists of a diagram using only the concepts and symbols of one language. So, each layer can be read easily, but models the relevant information. The combination of these two layers conveys the necessary information for the modelling of requirements for content recommendations.

Thirdly, the nature of our approach (the use of layers) allows us to extend the scope of the models without any difficulty. On the one hand, other languages can be used together via layers. On the other hand, we can imagine adding more details to our models by adding other layers, in order to manage more aspects of a RS for OSNs.

The main limitations of our models are related to the dynamics. Firstly, our diagrams show "one instance" of the mechanism. However, the fact that we can observe n instances of the mechanism could be modeled more clearly. Secondly, we show the interaction between two users. However, on OSNs many users are involved in many different mechanisms. This situation does not show on our models here.

Also, the distinction between user roles is limited to what they do. We could also take into account their motivations in order to distinguish between them.

Finally, we did not analyze scaling problems associated to the graphical notation.

5 Related Work

There is considerable work on extending requirements modeling languages. Souza et al. [3, 4] proposed the AwReqs and EvoReqs. AwReqs represent undesirable

situations to which stakeholders would like the system to adapt in case they happen. They can be used as indicators of requirements convergence at runtime. EvoReqs prescribe what to do in these situations. They consist of specific changes to be carried out on the requirements model, under specific circumstances. They are modeled as Event-Condition-Action (ECA) rules that are activated if an event occurs and a certain condition holds. Ali et al. [5] addressed the reasoning with contextual requirements. The connection between the various concepts, that is, the AwReqs and EvoReqs for the proposal of Souza et al. [3, 4], and the contexts for Ali et al. [5] is not as integrated as ours. For the former solution, Souza et al. [3, 4] model the AwReqs on the requirements model, but the EvoReqs are rules specified in natural language outside of the diagram. For the latter solution, Ali et al. [5] clearly manage the context, and integrate this notion directly in the models. However, the notion of dynamics is not included, whereas we propose an integrated solution, including the dynamics aspect. Other examples of extensions are discussed in the longer version of this paper [1].

6 Conclusion and Future Work

We believe i* is appropriate for the modeling of OSN requirements. However, as mentioned above, the existing concepts in i* do not allow us to model the dynamics observe in the use of OSNs. Hence, we proposed an add-on to the existing framework, by introducing a second layer. The latter consists of a Petri Net modelling the dynamics observed in OSNs.

Future work will consist in addressing the limitations we raised in Section 4. More specifically, we aim at providing a more general model, taking into account the various mechanisms an individual user can be involved in; as well as the several instances of mechanisms that can exist. We will also try to address scaling issues as well as to take into account the various motivations of different users.

References

1. Bouraga, S., Jureta, I., Faulkner, S.: Modelling requirements for content recommendation systems. arXiv:1606.05746 (2016)
2. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) (1989) 541–580
3. Souza, V.E.S., Lapouchnian, A., Mylopoulos, J.: (requirement) evolution requirements for adaptive systems. In: Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE Press (2012) 155–164
4. Souza, V.E.S.: A requirements-based approach for the design of adaptive systems. In: Proceedings of the 34th International Conference on Software Engineering, IEEE Press (2012) 1635–1637
5. Ali, R., Dalpiaz, F., Giorgini, P.: Reasoning with contextual requirements: Detecting inconsistency and conflicts. Information and Software Technology **55**(1) (2013) 35–57