Perspectives on User Story Based Visual Transformations

Yves Wautelet[1], Samedi Heng[2], and Manuel Kolp[2]

[1] KU Leuven, Belgium
yves.wautelet@kuleuven.be,
[2] LouRIM, Université catholique de Louvain, Belgium
{samedi.heng, manuel.kolp}@uclouvain.be,

Abstract. This paper summarizes previous works done by the authors on User Story (US) template unification and visual requirements models generation out of a US set. Indeed, transformation of a US set tagged using templates from a unified model to a Goal-Oriented model called the Rationale Tree and to a UML Use-Case Diagram are previous contributions summarized here. It also introduces the genuine contribution of generating a UML class diagram from a US set. Future research – notably on the use of the transformations in real life-case studies – is also discussed. Finally, the CASE tool supporting the approaches is overviewed.

Keywords: User Story; Rationale Tree; Use-Case Diagram; Agile Development

1 Introduction

User stories are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system [2]. However, no unification is provided in *User Story* (*US*) templates [9]. Indeed, the general pattern relates a WHO, a WHAT and possibly a WHY dimension, but in practice different keywords are used to describe these dimensions (e.g. Mike Cohn's *As a <type of user>, I want <some goal> so that <some reason>* [2] which could be instantiated to *As a <DRIVER>, I want to <register to the service> so that <I can propose a ride to go from A to B>*; a series of examples can be found in [10]). Moreover, in the literature, no semantics has ever been associated to these keywords. This is why, [9] conducted research to find the majority of templates used in practice, sort them and associate semantics to each keyword. These semantics were derived from several sources and frameworks; some of these are derived from Goal-Oriented Requirements Engineering (GORE, see [4]). The research lead to build a unified model of US templates with only a minimal but sufficient amount of keywords; most of the semantics adopted for these keywords were selected from the i* framework (i-star [11, 3]) The entire research process can be found in [9] while Section 2 summarizes the US templates unified model.

One may question the utility of such a model; why should US be "tagged" to a certain template. The main advantage is that, if the tagging respects the semantics associated to the concepts, it provides information about both the nature and the granularity

of the US element. Even in an agile context, this is useful for performing requirements analysis [5]. This paper summarizes the transformations from a tagged US set to a GORE model called the Rationale Tree (from [10]) as well as to a UML class diagram (from [8]). It also overviews future work around the Rationale Tree. Last but not least, preliminary work around the transformation from a tagged US set to a UML class diagram is depicted; this constitutes a genuine contribution of this paper.

2 Unified-Model of User Stories' Descriptive Concepts

Figure 1 represents the meta-model of US templates built in [9] in the form of a class diagram. The instance of one of these classes is a US element in itself from a concrete US. A US template can be designed taking an element from the WHO, WHAT and possibly WHY dimensions. The link between the classes conceptually represents the link from one dimension to the other. Specifically, the unidirectional association from the *Role* to one of the *Capability*, *Task* or *Goal* classes implies that the target class instantiates an element of the WHAT dimension (always tagged as *wants/wants to/needs/can/would like* in the model). Then, the unidirectional association from one of these classes instantiating the WHAT dimension to one of the classes instantiating the WHY dimension (always tagged as *so that* into the model) implies that the target class can instantiate an element of the WHY dimension. The following is a US template supported by our model: *As a <Role>, I would like <Task> so that <Hard-goal>*.

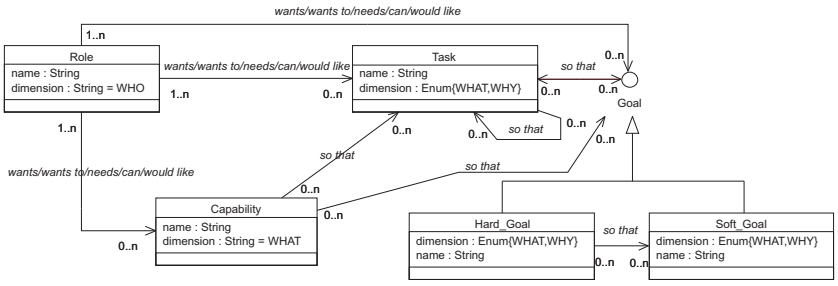


Fig. 1. Unified Model for User Story Descriptive Concepts (from [9]).

Each concept is associated with a particular syntax (identical to the name of the class in Figure 1) and a semantic. Due to a lack of space we do not depict the semantic associated to each of the concepts here; it can be found in [9, 10].

[9] pointed out the need of 3 possible levels of granularity within the functional elements expressed in US. We thus distinguish the *Hard-goal*, *Task* and *Capability*.

The *Hard-goal* is the most abstract element; there is no defined way to attain it and several ways could be followed in practice. It is indeed part of the problem domain. The *Task* represents an operational way to attain a *Hard-goal*. It is thus part of the solution domain. An example of a *Hard-goal* could be to *Be transported from Brussels to Paris*;

it can be the *Hard-goal* of a traveler but there are several ways to attain this *Hard-goal* (by train, by car, etc.).

The *Task* and the *Capability* represent more concrete and operational elements but these two need to be distinguished. The *Capability* does in fact represent a *Task* but the *Capability* has more properties than the former since it is expressed as a direct intention from a role. In order to avoid ambiguities in interpretation, we point to the use of the *Capability* element only for *an atomic Task* (i.e., a task that is not refined into other elements but is located at the lowest level of hierarchy). A *Task* could then be *Move from Brussels to Paris by car* and a *Capability* would be *Sit in the car*.

In practice, US elements need to be compared to each other to properly determine their type. Since this is also subject to interpretation these elements are re-tagged several times when they analyzed and structured.

3 From User Story Set to Rationale Tree: Goal-Based Approach

Building the Rationale Tree from a User Story Set. Visual GORE models were envisaged for graphical representation in [10]. From this [10] develops a decomposition structure for US elements called the Rationale Tree, largely inspired by i*. The icons used for its representation are illustrated in Figure 2.

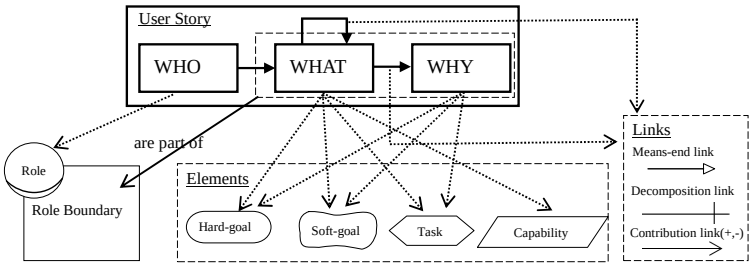


Fig. 2. Elements of the Rationale Tree: Graphical Representation (from [10]).

Figure 3 shows a Rationale Tree both in canonical form and instantiated to the carpooling example (see [10] for the US set). The US including the *Task* "*Pay by SMS in domestic country*" and the US including the *Task* "*Pay by credit card*" are *Epic* US (i.e. a US too abstract (coarse-grained) to be estimated, implemented and tested at once) because these are top-level *Tasks* related to means-end decompositions of the *Hard-goal* "*Pay for the car pooling service in function of the country he is traveling in*".

Rationale Tree: Benefits and Future Perspectives. The interest of the Rationale Tree approach essentially lies in the possibility to use a tool that supports reasoning within the requirements set initially expressed through US. The decomposition of elements helps to evaluate tactics for Hard-goal and Task fulfillment as well as requirements consistency. Missing (or missing parts of) requirements can thus be identified. Reasoning can also help provide justifications for architectural choices made for the support of Soft-goals.

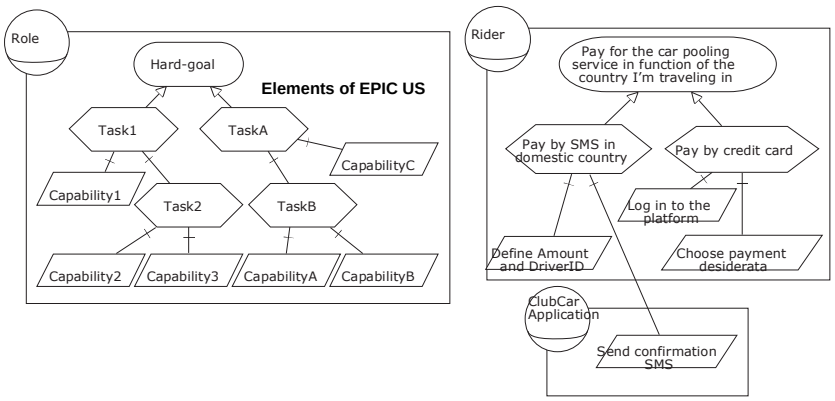


Fig. 3. Top-Level Hard-goal, Several Means-End Decompositions (from [10].

Real life case studies are currently being performed using the Rationale Tree approach in SCRUM projects. It has so far notably been applied for the reinterpretation of the requirements in a travel and expense management application. As first result we highlight that visual identification of the requirements dependencies allows a more efficient re-usability of elements developed in the project. This case goes further than a simple application of the rationale tree and also integrates the latter directly in the SCRUM board. Then, coupled to an algorithm for determining elements with highest business value, iteration planning can be performed. This approach allowed to increase traceability and visibility on requirement elements across iterations and monitor the progress on multiple levels (i.e. the levels of the elements in the tree).

Future work also includes comparing the Rationale Tree Approach with Natural Language Processing (see [7, 6]) for discovering missing requirements in a US set.

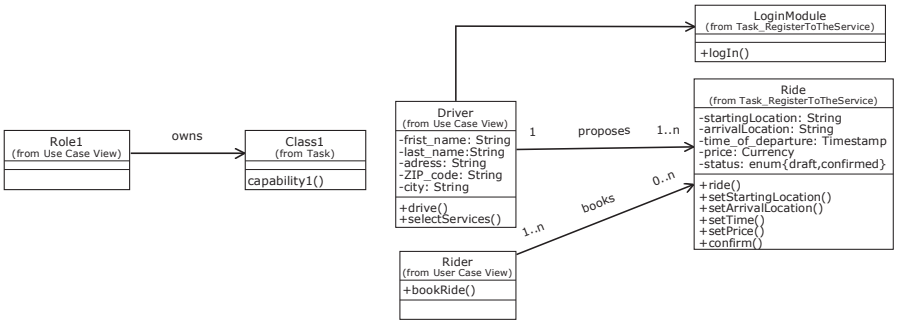
4 From a User Story Set to UML Diagrams

Building a Use Case Diagram from a User Story Set. Starting from a tagged set of US, [8] suggests a method to systematically build a UML *Use-Case Diagram* (*UCD*); Table 1 summarizes the mapping of elements. The UCD proposes a view focusing on US containing coarse-grained process elements (*Epic US*). This furnishes an abstract view of the system-to-be and helps with the scoping of US realizations.

Building a Class Diagram from the US Set and the Use Case Diagram. In future work, we will formally define forward engineering mechanisms from a US set to a UML Class Diagram. We can already highlight that Roles can be forward engineered into classes. Realization analysis of Goals and Tasks allows to identify more classes. In turn, Capabilities lead to class operations. This is shown in a canonical form in the left side of Figure 4 and instantiated on the Carpooling example in the right side of the figure. For example, the Task *Register to the service* from US3 requires to architecture the *Ride* class because it needs to create and manipulate *Ride* objects. The Capability *Confirm the proposal* from US5 leads to the operation *confirm()* of the class *Ride*.

Table 1. Mapping a US set with the UCD

US Set Element	*UCD Element*
Role	Actor
Hard-goal	Use Case; several Use Cases transformed from Hard-goals can be linked through <<*include*>> dependencies
Task	(Possible) Use Case; the Use Case transformed form a Task should be linked through <<*include*>> or <<*extend*>> dependencies with Use Cases transformed from Hard-goals
Capability	No possible transformation
Soft-goal	RUP/UML Business Goal



(a) Class Diagram Forwarded on the Basis of Stereotyped User Stories

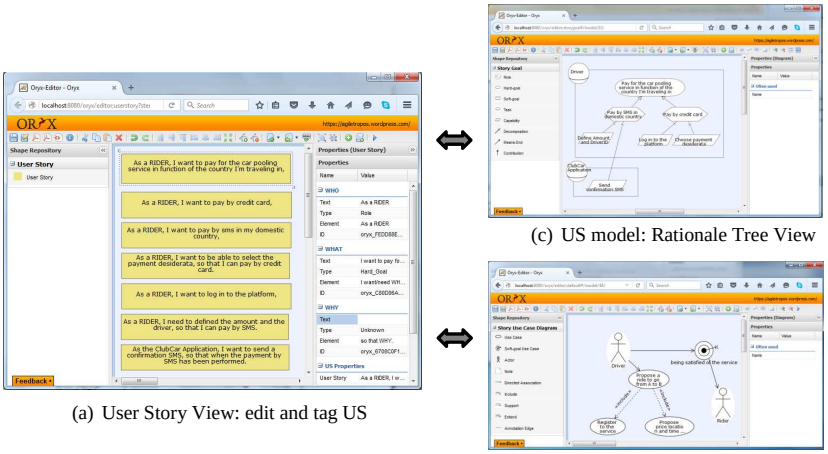
(b) Partial Class Diagram for the Carpooling Example

Fig. 4. Class Diagram: Canonical Form and Carpooling Example.

5 CASE-Tool and Conclusion

Automating Approaches and Round-Tripping Between Views. We have built an add-on to our Descartes CASE-Tool [1] to support transformations (see Figure 5). It allows multiple views: the *User Story View* edits US through virtual US cards; the *Rationale View* rationale trees; the *Structural View* structural agent diagrams; the *Use-Case View* a UCD and finally the *Class, Sequence and Activity Diagram Views*. The CASE-Tool synchronizes other views when changes are made. The editing process is continuous over the requirements analysis stage and he entire project life cycle.

Conclusion. US are popular informal artifacts for quickly expressing requirements in the agile methods. Through multiple contributions, we have shown that with little effort it is possible to bring more formality to the US sorting process and build a visual representation. These can be driven by GORE frameworks or the UCD. These visual representations are useful for building a high level picture of the system-to-be. The Rationale Tree also allows to study dependent requirements with an impact on re-usability or identify missing ones. Put in an iterative perspective, it allows business value based iteration planning and to monitor the project progress on multiple levels. Finally, we overviewed how a class diagram can be forward engineered out of a tagged US set.



(a) User Story View: edit and tag US

(c) US model: Rationale Tree View

(b) US model: Use-Case View

Fig. 5. The supporting CASE-Tool.

References

1. Descartes architect (2017), `http://www.isys.ucl.ac.be/descartes/`
2. Cohn, M.: Succeeding with Agile: Software Development Using Scrum. Addison-Wesley Professional, 1st edn. (2009)
3. Dalpiaz, F., Franch, X., Horkoff, J.: istar 2.0 language guide. CoRR abs/1605.07767 (2016)
4. van Lamsweerde, A.: Goal-oriented requirements enginering: A roundtrip from research to practice. In: 12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan. pp. 4–7. IEEE Computer Society (2004)
5. Liskin, O., Pham, R., Kiesling, S., Schneider, K.: Why we need a granularity concept for user stories. In: Proceedings of XP'14, Rome. LNBIP, vol. 179, pp. 110–125. Springer (2014)
6. Lucassen, G., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Visualizing user story requirements at multiple granularity levels via semantic relatedness. In: Conceptual Modeling - 35th Intl. Conference, ER 2016, Proceedings. LNCS, vol. 9974, pp. 463–478 (2016)
7. Robeer, M., Lucassen, G., van der Werf, J.M.E.M., Dalpiaz, F., Brinkkemper, S.: Automated extraction of conceptual models from user stories via NLP. In: 24th IEEE International Requirements Engineering Conference, RE 2016. pp. 196–205. IEEE (2016)
8. Wautelet, Y., Heng, S., Hintea, D., Kolp, M., Poelmans, S.: Bridging user story sets with the use case model. In: Link, S., Trujillo, J. (eds.) Advances in Conceptual Modeling - ER 2016 Workshops Proceedings. LNCS, vol. 9975, pp. 127–138 (2016)
9. Wautelet, Y., Heng, S., Kolp, M., Mirbel, I.: Unifying and extending user story models. In: CAiSE 2014, Thessaloniki, Greece. Proc. LNCS, vol. 8484, pp. 211–225. Springer (2014)
10. Wautelet, Y., Heng, S., Kolp, M., Mirbel, I., Poelmans, S.: Building a rationale diagram for evaluating user story sets. In: Tenth IEEE International Conference on Research Challenges in Information Science, RCIS 2016. pp. 1–12. IEEE (2016)
11. Yu, E.: Modeling Strategic Relationships for Process Reengineering, chap. 1–2, pp. 1–153. MIT Press, Cambridge, USA (2011)