Systems Engineering Requirements: A Guided Example of an Applied Knowledge System

Anabel Fraga and Juan Llorens

Carlos III of Madrid University, Madrid, Spain

Abstract. Knowledge is centric to systems engineering, the knowledge management process must take into account that a System Knowledge Repository (SKR) exists as a key element for either quality improvement, traceability support and, in summary, reuse purposes. Requirements engineering in the Systems Engineering process is enhanced by using knowledge systems and quality of requirements enriched as well. The more correct, complete and consistent a requirement is, the best performance it will have and knowledge systems enable a more exhaustive and fast quality process. A knowledge management process is proposed and it is guided by a requirements domain based example.

Keywords. Requirements Engineering, Ontologies, Knowledge, System engineering, Reuse.

1 Introduction

The application of ontology engineering in systems engineering seems to be a very promising trend [12], [4]. We call it system verification based on Knowledge Management, and it deals with assisting System Engineers to get a complete and consistent set of requirements (e.g. compliance to regulation, business rules, non-redundancy of requirements…) by using Ontologies, which represent the domains of knowledge of an organization. The combination of Requirements Engineering with Knowledge Management, throughout Information Retrieval from existing sources, allows the verification process to measure quality of a set of requirements by traceability, consistency/redundancy, completeness and noise. Information retrieval enables also to verify the completeness of the ontology using a PDCA (Plan-Do-Check-Act) cycle of improvement. Requirements engineering is the first step and by traceability and Ontology based systems, similar assets of any phase of the development process used in analogous projects could be reused and adapted to a new challenge. For instance, by using a semantic approach, a requirement can be translated into a graph by means of NLP (Natural Language Processing) techniques.

In order to build a knowledge repository for managing requirements quality, the first activity must be to clearly define the typology of requirements that are going to be covered by the knowledge system, because this typology will affect the requirements structure and vocabulary. For example, an organization would be interested in managing the quality of "performance requirements". Even if it seems to be simple, the selection of the kind of requirements to formalize in a knowledge repository is

Fraga A. and Llorens J..
Systems Engineering Requirements: A Guided Example of an Applied Knowledge System.
DOI: 10.5220/0005177700190030
In *Proceedings of the 5th International Workshop on Software Knowledge* (SKY-2014), pages 19-30
ISBN: 978-989-758-051-2

not trivial. "In order to define what I want, I usually need to know what I do not want". And this is a real problem. Because in many cases "performance requirements" will NOT be considered by you as "functional requirements", but in other cases they certainly will. Figure 1 shows a requirement types taxonomy.

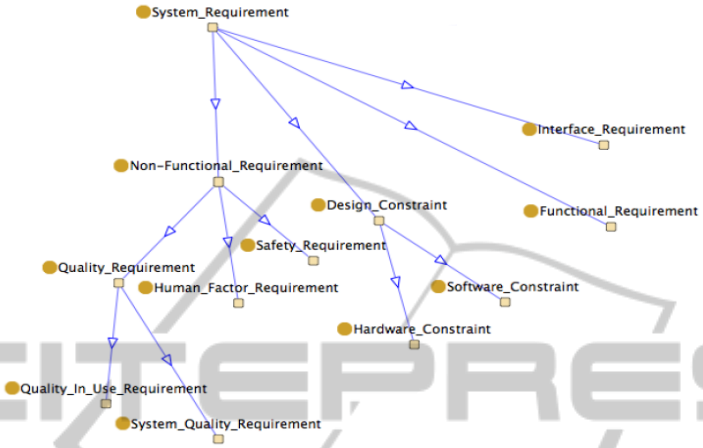


Fig. 1. Taxonomy of requirements types.

A requirement is an identifiable element of a function specification that can be validated, and against which an implementation can be verified. [1] [7] [6] [11] [5] [3] [2] Requirements Development encompasses all of the activities involved in eliciting, analyzing, specifying and validating the requirements.

Requirements Management encompasses the activities involved in requesting changes to the requirements, performing impact analysis for the requested changes, approving or disapproving those changes, and implementing the approved changes. Furthermore, includes the activities used for ensuring that work products and project plans are kept consistent and tracking the status of the requirements as one progresses through the software development process.

There are some rules that establish how the requirements must be written and which mistakes must be avoided. The INCOSE (International Council on Systems Engineering) rule states that the requirements must be clear and concise, complete, consistent, verifiable, affordable, achievable, necessary, bounded, traceable with independent implementation. [8] Any mistake in the requirements definition phase is distributed downwards until low level requirements being almost impossible to fix. Thereby, those mistakes must be caught in the early development process.

2 Knowledge Management Process

Stage 0. Inmature Situation

An organization is in an Immature situation for the knowledge management process when there is no conscious understanding of the need of managing assets for

their (re)-use internally.

Stage 1: Managing SAS

This stage is gathered as soon as the organization has implemented whatever activity for supporting an assets store. Usually, this store is based on a defined model for an assets repository (the most common is the OMG [13]).

Stage 2: Managing Terminology

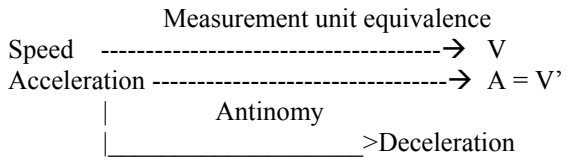
This stage is reached when the organization is managing the terminology that affects the system of interest. The existence of a controlled vocabulary of terms, as concepts, allows the quality management process to produce relevant metrics about the work-products, and therefore it is a key stage to provide qualified assets as well.

In this stage we will start with our guided example, imagine a system "Car", with components like "Brakes", "Pedals", and also actions even when the pedals are "pressed", and also additional variables as "velocity" and "time of speed".
Here the vocabulary and thesaurus must be built. For instance:

The project breakdown structure (PBS) also contains useful information:

Car = System
Brake = Brake System
Pedal of the brake
Pedal States: Pressed, Released

Knowledge about the physical world also helps to understand the requirement:



Stage 3: Managing the System Conceptual Model (SCM)

When an organization supports a persistent representation of a SCM, it becomes an asset ready to be reused. The existence of a SCM allows the quality management process to produce advanced metrics about the work-products. In organizations that only want to develop the Knowledge Management Process for Knowledge Reuse, this stage represents the possibility to reuse "ground truth" knowledge at the organization level. Those organizations that do not want to develop a SCM can have a light stage, with almost no effort.

In this example the organization manages assets for indexing and retrieval. In this stage the semantic grouping of concepts must be managed, at conceptual level it is not showed but semantics will be implemented in a tool, as for instance Protégé [10], Knowledge Manager [9], and so on.

Stage 4: Managing Patterns

Patterns are developed with the intention to be matched to Systems Engineering work-products content. When an organization owns a set of patterns, it becomes possible to identify relevant structures inside the produced content. By using patterns matching, specialized software can identify the patterns inside artifacts and be able to assign them the patterns as a means to classify them. The existence of patterns allows the quality management process to identify the typology of the work-products. On the other side, the Knowledge Management process needs patterns as the first stage to properly formalize them.

In the case of a requirement like: *"Whenever the pedal of the brake is pressed, the car shall decelerate immediately"*

A quick first syntactical analysis will present the following structures:

Whenever
The
Pedal of the brake
Is pressed
The
Car
Will
Decelerate
Immediately

Two syntactic structures have to be considered here: The compound noun "pedal of the brake" and the compound verb "is pressed". The joining of those words as a single compound phrase is performed by the Tokenization stage of the process.

We´ll not produce patterns at syntactical level because we consider that "Pedal of the brake" is, in itself, a term of the domain with full meaning. This means that "Pedal of the brake" must be included in the ontology. The structure "is pressed" will be split in the two terms that form it: the verb "to be" and the verb "to press" in participle.

The patterns structure looks like:

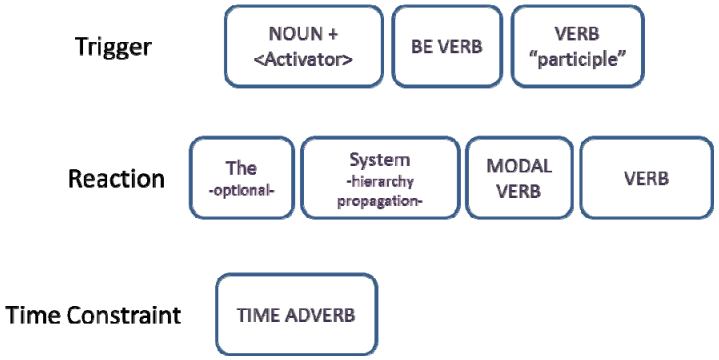


Fig. 2. Pattern structure for the guided example.

Stage 5: Managing Formalization

In stage 4 the organization owns an ontology with formalization capabilities for different work-product types. The formalization information, today, is formed by rules for producing formal representations of work-products.
The existence of Formalization information allows the quality management process to produce advanced metrics about the work-products.
Due to the existence of formalization capabilities, the supported operations in the previous stage are improved in the following way:

- Artifact retrieve
  The capability of the repository to formalize artifacts allows producing smart retrieval algorithms for semantic search, improving precision.
- Artifact traceability
  Due to the formalization capability, automatic trace policies based on similar content can be produced.

The following formalization will be used for the example we follow in this paper:

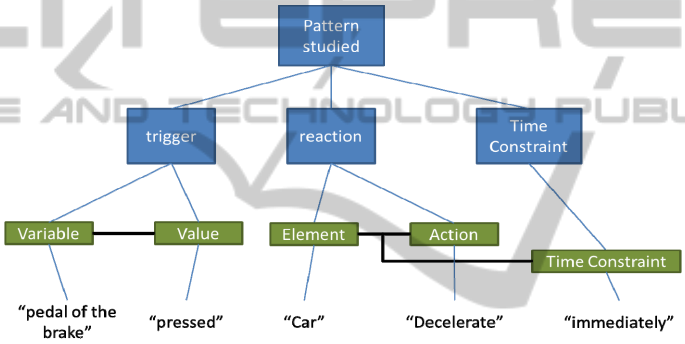


Fig. 3. Formalization of the requirement.

As it can be seen in the previous diagram, the trigger pattern produces by itself a graph:

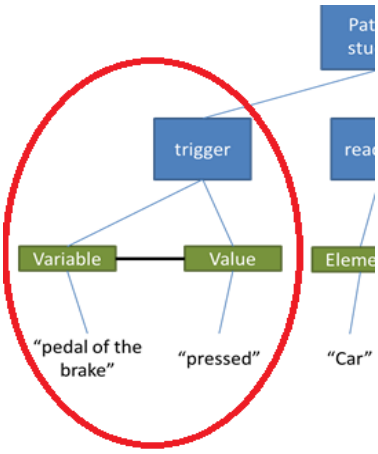


Fig. 4. Graph of the trigger.

The complete representation will be:

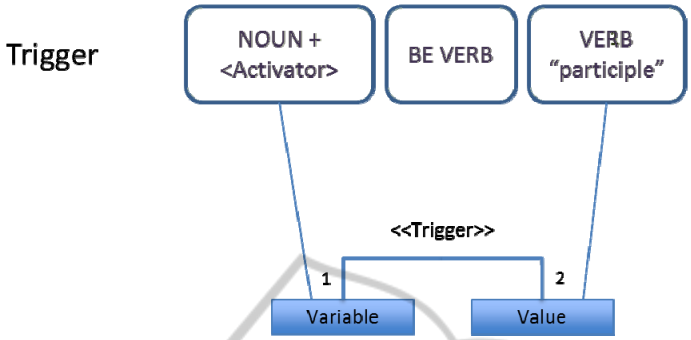


Fig. 5. Example of formalization.

But, a problem arises when trying to formalize the graphs out of the other patterns: the graph we want to produce is formed by elements coming from different patterns (reaction and time constraint):

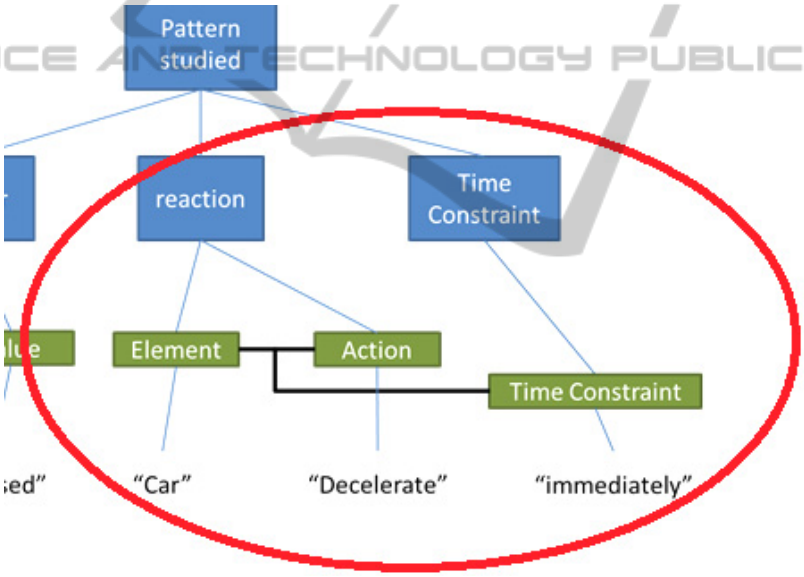


Fig. 6. Example of formalization #2.

This means that it is not possible to formalize the graph only at the Reaction pattern, and also not possible to formalize it at the time constraint pattern.
There are two solutions to this problem:

- Unify the reaction pattern to include a section for the time constraint.
- Create the graph at the next level, where both patterns exist as sub-patterns.

In our case, we´ll use the second solution, so the formalization will be produced at the next level of patterns.

This formalization process is, of course, very human dependent, and we must not be afraid of that. The result of this process must be the way the organization understands the requirement semantics. Just by defining a way, good benefits of it can be gathered (sharing same understanding, promoting the view to the supply chain etc etc.)

For example, other Systems Engineering group could see the studied requirement as a state machine transition, and the modeling could be a different one.

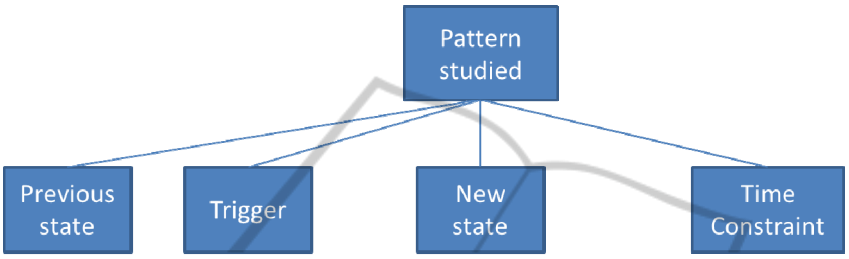


Fig. 7. Pattern structure.

Stage 6: Managing the System Knowledge Repositiry (SKR)

In stage 5 the organization has included inference information to the ontology, completing the System Knowledge Base (SKB) and forming a complete System Knowledge Repository (SKR).

Inference information is necessary, as well, for the quality management process, in order to produce consistency metrics.

Usually, the inference rules for decision making purposes are based on the existence of a full developed SCM. Therefore, in some cases, the SCM is fully developed in this stage.

Due to the existence of inference capabilities, the supported operations in the previous stage are improved in the following way:

- Artifact transformation

The existence of inference information allows generating new knowledge based on previous artifacts. For example: a rule could allow a user to state a complex query using UML, asking the repository to create new UML models with the information in the similar artifacts that is NOT existing in the query, and being notified of them by producing new artifacts.

- Artifact traceability

Due to the inference capability, automatic trace policies based on transformations can be produced.

Once a terminology, a thesaurus, patterns and its formalization has been done, if a requirement instance of a type matching is needed, as for instance: “If the car is moving, whenever the pedal of the brake is pressed, the car will decelerate immediately”. The SCM plays its role and indexing/retrieval systems are involved in the process. The knowledge system is completed and implemented in a Knowledge System as Knowledge Manager [9] that fully support this kind of process and even patterns are available for this kind of requirements.

3 How to Manage Terminology: a Basic Step for the Process

As stated for Stage 2, the existence of a controlled vocabulary of terms, as concepts, allows the quality management process to produce relevant metrics about the work-products, and therefore it is a key stage to provide qualified assets as well. The sequence diagram explaining the process is shown in Figure 8. First of all, it is important to notice that diverse environments (areas) are suggested to be active in any new implantation of a Knowledge Management Process: Production, Pre-production, and Evolution (New Version of the Ontology). In the Evolution area, the ontology is evolved based on the domain artifacts once it is ready then it is set as active and replicated in Pre-Production and Production areas. In the Pre-Production area indexing of new artifacts in an incremental mode is done based on the ontology active at that moment that maybe is not the one active in the Evolution area. In the Production area the retrieval process is done based on the indexing available and the ontology active at that point. Figure 8 shows a piece of the process between Production and Evolution areas focus on the terminology point of view.

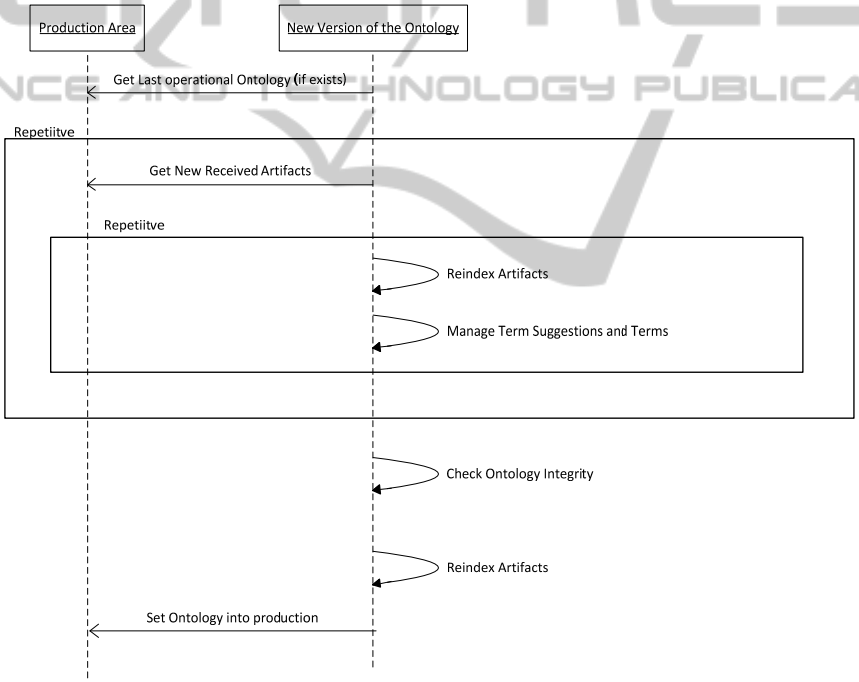


Fig. 8. Terminology process sequence diagram.

A Terminology List is needed for standardizing and normalizing the terminology used in the custom application. The input information must/should match the controlled vocabulary. Using a glossary with different categories of terms, the ontology may store:

- Business related Terms: those terms central to the business area(domain) to be treated.

- General Language Terms: those terms related to the basic language used in the ontology (i.e. English, Spanish, German, and so on).
- Syntactically relevant phrases: those are general language terms related to the General Language Terms: Adverbs, Adjectives, etc.
- Stop Terms: those terms that could be of no relevance.

Gathering Vocabulary: How It Works

The terminology of the domain must be extracted of the artifacts in the domain. For that reason, a sequence of steps must be completed, as shown in Figure 9:

- Tokenization
- Normalization
- Disambiguation
- Phrase recognition

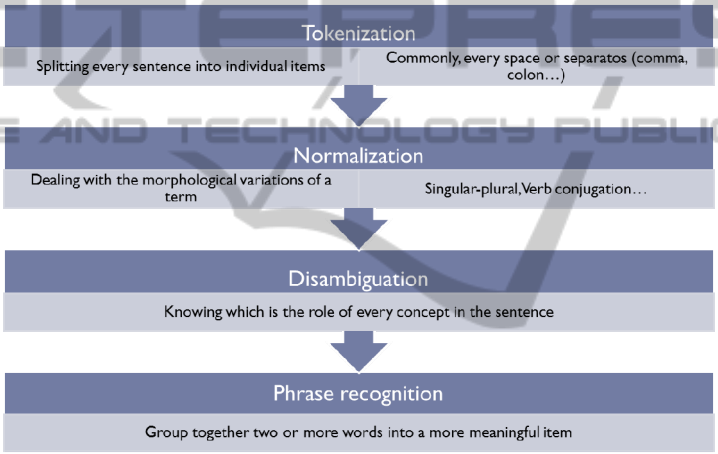


Fig. 9. Steps for gathering terminology.

An example of the steps to be completed for gathering the terminology of the domain is as follows:

All Radars shan't identify the following targeting enemy objectives:

Tokenization	[All] [Radars] [shall] [not] [identify] [the] [following] [targeting] [enemy] [objectives][:]
Normalization	[All] [Radar] [shall] [not] [identify] [the] [following/follow] [targeting/Target] [enemy] [objective] [:]
Disambiguation	[All] [Radar] [shall] [not] [identify] [the] [following] [Target] [enemy] [objective] [:]
Phrase Identification	[All] [Radar] [shall] [not] [identify] [the] [following] [Target] [enemy objective] [:]

The steps of tokenizing, normalizing, disambiguation and phrase recognition are shown above. As complementary, a state machine diagram is included as Figure 10 as an illustrative sequence of the whole process.

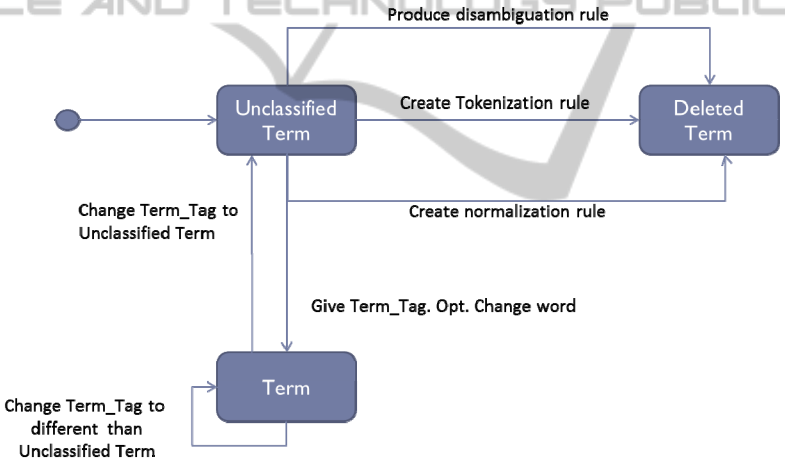


Fig. 10. State diagram showing the gathering terminology steps.

Ontologies Maintenance: Effort / ROI Perspective

The perception that building ontologies to improve Systems Engineering could imply a project with negative ROI is certainly real within organizations. The REUSE Company has produced a methodology that has as main goal to discourage this thinking. The fundamental aspects of this approach are:

- Define a SIMPLE schema/ Meta-model for the Ontology, avoiding complex structure
- Create Semi-automatic methods to produce the ontology layers at low cost

- Define clear applications of the ontology inside the SE lifecycle, like, for instance, the requirements quality control
- Define a PDCA cycle to reduce the cost of maintenance

Finally, the experience has shown that, once the ontology is specialized to particular usages, the maintenance cost decreases very quickly once a determined level of maturity is gathered.

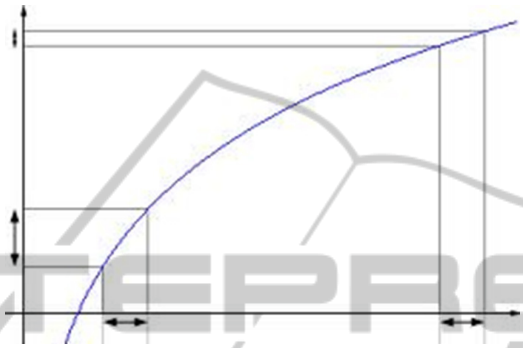


Fig. 11. ROI graphical representation.

These arguments allow producing positive ROI easily.

4 Conclusion

Once the Knowledge Management Process was introduced and a guided example shown, the next step is the full definition of the example in a tool supporting all the Lifecycle of a requirement coexisting in a Knowledge System Repository. It is in progress and partially implemented at this point. The whole tool is in its process to be improved and completed for supporting the whole Knowledge Management Process for any project, company or organization.

The requirements in any project are one of the most important assets, if not the most. A bad group of requirements might have terrible implications in a developed system. For instance a requirement detailed in various parts of the requirement list using different measurement units might cause an accident or a failure during operation of any system.

Classical sequential review process of requirements is costly in terms of time consuming. Then support of tools for lexical, syntactic analysis enables to correct bad requirements writing before business of project reviews.

One of the next challenges in the Industry is to reach an ontology-assisted system engineering process to write SMART requirements at a first shot.

The use of ontologies and patterns is a promise of doing better requirements engineering and knowledge reuse in any system engineering project.

Acknowledgements

Authors thank Manuela Berg and Thomas Peikenkamp, both from OFFIS, for the requirement textual example provided for this guided definition of the process.

The research leading to these results has received funding from the european union's seventh framework program (fp7/2007-2013) for crystal – critical system engineering acceleration joint undertaking under grant agreement n° 332830 and from specific national programs and / or funding authorities.

References

1. Eric Braude. Software Engineering. An Object-Oriented Perspective. John Wiley & Sons, 2001.
2. Genova, G. Fuentes, J. Llorens, J. Hurtado, O. Moreno, V. A framework to measure and improve the quality of textual requirements, Requirements Engineering, Springer, Url: http://dx.doi.org/10.1007/s00766-011-0134-z, Doi: 10.1007/s00766-011-0134-z
3. Guide for Writing Requirements, INCOSE Product INCOSE-TP-2010-006-01, V.1, April 2012
4. H. Chale et al., Reducing the Gap between Formal and Informal Worlds in Automotive Safety-Critical Systems, INCOSE Symposium 2011, Denver.
5. Ian F. Alexander & Richard Stevens. Writing better requirements. Addison-Wesley, 2002.
6. Ian Sommerville & Pete Sawyer. Requirements Engineering: A Good Practice Guide. John Wiley & Sons. 1997.
7. Ian Sommerville. Software Engineering. Pearson-Addison Wesley, 2005.
8. INCOSE, Systems Engineering Handbook. http://www.incose.org/ProductsPubs/products/sehandbook.aspx (last visited 11/13/2013)
9. OBSE Fundamentals, The Reuse Company, Juan Llorens (UC3M) - José Fuentes (TRC). http://www.reusecompany.com (last visited 11/13/2013)
10. Protégé. Stanford University Development. http://protege.stanford.edu (last visited 11/15/2013)
11. Roger Pressman. Software Engineering: a practical guide, 6ª ed. McGraw-Hill.
12. The Reuse Company (TRC). http://www.reusecompany.com (last visited 11/13/2013)
13. The Reusable Assets (RAS) - OMG. http://www.omg.org/spec/RAS/2.2 (last visited 01/07/2014)