What is essential? – a pilot survey on views about the requirements metamodel of reqT.org

Regnell, Björn

Published in:
Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)

DOI:
10.1007/978-3-319-30282-9_16

2016

Document Version:
Peer reviewed version (aka post-print)

What is essential? – A pilot survey on views about the requirements metamodel of reqT.org

Björn Regnell

Dept. of Computer Science, Lund University, Sweden
bjorn.regnell@cs.lth.se

Abstract. [**Context & motivation**] This research preview paper presents ongoing work on the metamodel of a free software requirements modeling tool called reqT that is developed in an educational context. The work aims to make an initial validation of a survey instrument that elicits views on the metamodel of the reqT tool, which aims to engage computer science students in Requirements Engineering (RE) through an open source DSL embedded in the Scala programming language. [**Question**] The research question is: Which RE concepts are essential to include in the metamodel for a requirements engineering tool in an educational context? [**Principal ideas**] A survey instrument is developed, with a list of 92 concepts (49 entities, 15 relations and 28 attributes) and a set of questions for each concept, to elicit the respondents' views on the usage and interpretation of each concept. [**Contribution**] The survey is initially validated in a pilot study involving 14 Swedish RE scholars as subjects. The survey results indicate that the survey is feasible. The analysis of the responses suggest that many of the concepts in the metamodel are used frequently by the respondents and there is a large degree of agreement among the respondents about the meaning of the concepts. The results are encouraging for future work on empirical validation of the relevance of the reqT metamodel.

Keywords: requirements engineering, metamodel, CASE tool, engineering education, embedded domain-specific language, empirical software engineering.

1 Introduction

There are many challenges in teaching Requirements Engineering (RE) [4, 6], including advancing students' requirements modelling skills that can be used effectively in an unstructured, non-ideal, real-world situation [1]. When teaching RE modelling we may ask ourselves: What are the *essential* RE concepts that we should include in a taught metamodel for requirements? This paper investigates this questions in conjunction with the on-going work of developing a metamodel for reqT.org, an open source requirements engineering tool used in RE education [7]. A survey instrument is presented aiming to elicit the frequency of RE term usage and the degree of interpretation agreement. The responses from 14 Swedish RE scholars are analysed and discussed and conclusions suggest that a large subset of the concepts of the current reqT metamodel can be seen as "essential" in that a majority of the subjects use them while agreeing with the concepts' definitions. The presented work represents an initial validation of the survey instrument. Further work involving more subjects is needed to draw conclusions with more certainty.

2 Background

There are nowadays numerous commercial RE tools available, but many are expensive, complex and not sufficiently open [2]. A major aim of the reqT open source project is to provide a small but scalable, semi-formal and free software package for an educational setting [7] that can inspire code-loving computer science students to learn more about requirements modeling. The tool development started in 2011 at Lund University, where reqT is used in RE teaching at MSc level in student role-playing projects.[1]

A critical issue is how to choose the essential RE concepts that allows for sufficient expressiveness, while not overloading the metamodel with esoteric concepts just for the sake of completeness.

The reqT metamodel includes three types of concepts: entities, attributes and relations. Entities and attributes are nodes in a graph data structure, while relations are edges that can connect entities with sub-graphs. Thus a tree-like structure can be created of arbitrary depth spanning the graph that models some chunk of requirements.

The code below shows a toy example of an orthogonal variability model [5] expressed in the reqT Scala-embedded DSL [7] illustrating a small part of its metamodel. Other parts of the metamodel contains concepts that enable e.g. goal modelling, use case modelling, and user story modelling, see further Appendix A.

```
Model(
  Component("appearance") has (
    VariationPoint("color") has (
      Min(0), Max(2), Variant("blue"), Variant("red"), Variant("green")),
    VariationPoint("shape") has (
      Min(1), Max(1), Variant("round"), Variant("square")),
    VariationPoint("payment") has (
      Min(1), Max(2), Variant("cash"), Variant("credit")),
    VariationPoint("payment") requires Variant("cash"),
    Variant("round") excludes Variant("red"),
    Variant("green") requires Variant("square")),
  Component("appearance") requires VariationPoint("shape"),
  App("free") has Component("appearance"),
  App("free") binds (VariationPoint("shape") binds Variant("round")),
  App("premium") has Component("appearance"),
  App("premium") binds (
    VariationPoint("color") binds (Variant("red"), Variant("green")),
    VariationPoint("shape") binds (Variant("round"), Variant("square")),
    VariationPoint("payment") binds Variant("cash")))
```

Entities in the above code listing are in bold, attributes in italics and relations start with a lower case letter. In the reqT editor, entities, attributes, and relations are syntax-coloured in blue, green and red respectively. A reqT model written in the above syntax is actually valid Scala code that, when executed, generates a data structure that can be traversed and manipulated using Scala scripts. Visualisations can be generated by export to GraphViz. Export is also available to HTML and spreadsheet formats.

[1] The Lund Univ. MSc-level RE course can be found at: `http://cs.lth.se/education`

3 Methodology and Data Collection

In order to validate RE scholar's opinions of the metamodel, a survey instrument was developed including the 49 entities, 15 relations and 28 attributes. All concepts and definitions are listed in Appendix A. [2] The concepts were gathered from various sources including the IREB Glossary [3], Wikipedia, agile development, variability [5] and goal modelling, and the text book [3] used in an RE course at Lund Univ.[1]

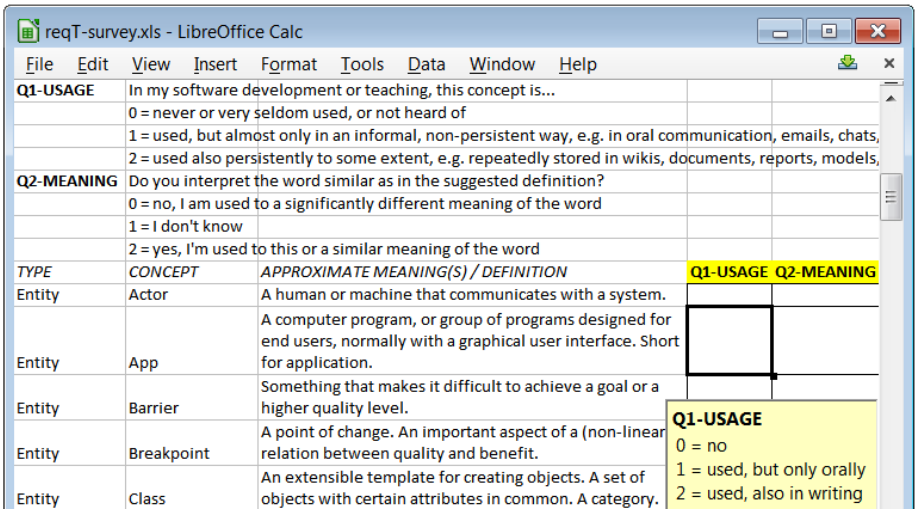

Fig. 1. A screen dump of a part of the survey instrument.

The data collection was made during a Swedish national network meeting with academic RE scholars in spring 2015. The survey was filled in during the meeting using the participants' own laptops in a spreadsheet shown in Fig. 1. The subjects were given around 20 minutes to complete the survey. Most of the subjects handed in the survey via email directly after the session, while a few finished it after the meeting.

4 Data Analysis

Subject background. The background questions in the survey regards the role of the subject, as shown in Table 1. The analyzed[4] total number of subjects is 14, of which 10 are teachers, 10 are developers and 13 are researchers. The response rate was 100% after a reminder was emailed to one missing subject.

Frequency analysis. The degree of "essentiality" is characterized as the number of subjects that has responded that they (1) use the concept at least in an informal, non-persistent way, *and* that they (2) use the concept in a similar meaning as in the definition in Appendix A. Fig. 1 shows the definitions of the three-level ordinal scales of Questions

[2] The survey is available at `https://github.com/reqT/reqT/tree/3.0.x/survey`
[3] `https://www.ireb.org/en/cpre/cpre-glossary/`
[4] One subject answered NO on all background questions and was therefore excluded.

$Q1_{usage}$ and $Q2_{meaning}$ respectively. Table 2 shows the results of the frequency counts. If an "essentiality threshold" is chosen at $N/2$ then only the 9 concepts from row $n = 7$ and below in Table 2 are considered "non-essential", hence showing that more than 90% of the metamodel concepts have a majority of the subjects that use them and agree upon their definitions. Each concept has at least one subject that uses it and agrees with its definition.

The following 19 concepts were reported "missing": S01: *or*, S02: *bug*, *threshold*, S04: *role*, *problem*, *motivates*, *and*, *or*, *pattern*, *submodel*, S06: *plug-in*, *informalism*, S07: *full sentence*, S09: *satisfaction*, *satisfies*, *customer*, S11: *system-of-interest*, *verification*, *validation*, S13: *context*. Thus, the concept 'or' was the only concept that had consensus among several subjects (S01, S04) as considered "missing".

The anonymised data and analysis scripts (developed using Scala and Apache POI) are available at: `https://github.com/bjornregnell/reqT-survey`

Table 1. Background of subjects, $N = 15$. The subjects were given anonymous ids S01–S15.

Background question	*Subject responding YES*
Do you teach software engineering and/or requirements engineering? YES/NO	S01 S03 S04 S05 S07 S08 S09 S11 S12 S14
Do you develop software by writing code and/or creating system models? YES/NO	S01 S02 S03 S07 S08 S09 S10 S11 S13 S14
Do you do academic research in software and/or requirements engineering? YES/NO	S01 S03 S04 S05 S06 S07 S08 S09 S10 S11 S12 S13 S14

Table 2. Frequency analysis, where n is the number of subjects that for the respective concept answered $(Q1_{usage} >= 1)$ *and* $(Q2_{meaning} = 2)$. In total there are 92 concepts (49 entities, 15 relations and 28 attributes). The higher up in the table, the more "essential". For $n = 0, 2, 3, 5$ the were no concepts with answers by that number of subjects.

n	*Entities*	*Attributes*	*Relations*
14	`Class, Component, UseCase, Variant`	`Comment, Example, Max, Min, Title`	`implements, verifies`
13	`Configuration, Data, Design, Event, Quality, Scenario, Stakeholder, System, Term`	`Code, Constraints, Cost, FileName, Probability, Profit, Spec, Why`	`excludes, interactsWith, is, relatesTo, requires`
12	`Actor, Domain, Feature, Function, Interface, Module, Relationship, Release, Req, Risk, Service, State, Task, Test`	`Benefit, Capacity, Frequency, Input, Order, Output, Prio, Text, Value`	`has, impacts`
11	`Idea, Label, Member, Meta, MockUp, Section, User`	`Image`	`precedes, superOf`
10	`Goal, Story`	`Expectation`	
9	`App, Issue, Target, WorkPackage`	`Damage`	`binds, helps`
8	`Item, Product, Resource, VariationPoint`		`deprecates`
7	`Breakpoint, Screen`	`Status`	
6	`Barrier`	`Deprecated`	`hurts`
4	`Ticket`		
1	`Epic`	`Gist`	

5 Discussion and Conclusion

The presented survey is a pilot investigation with two main contributions: (1) the survey instrument together with the data collection and analysis approach, which are shown to be feasible in the presented context, and (2) the pilot study results: for more than 90% of the 92 reqT metamodel concepts a majority of the 14 participating RE scholars claim to use them and agree upon their definitions. Only 1 concept was considered missing by more than one subject, while in total 19 additional concepts were reported missing by some subject.

Limitations. It can be questioned if "essentiallity" of a set of RE concepts can be characterized by how many RE scholars that use them and agree upon their definition, but it can also be argued that concept usage in an educational context is interesting to investigate when developing a metamodel for an academic RE tool. A major threat to external validity is the limited number of subjects. Due to few subjects and the high degree of homogeneity among subjects with respect to background, it is difficult to analyse and draw conclusions e.g. about potential differences in opinions between e.g. teachers and developers. Some subjects needed more time and completed their survey offline, which may give a variation in how carefully the responses were considered.

Further work. When developing a metamodel it is interesting not just to ask if the concepts to include are essential, but also to pose the question if the set of concepts is complete. If some essential concept is missing from some stakeholder's viewpoint, then the metamodel is not sufficient. With more subjects participating in the presented RE metamodel survey, the analysis of answers to further questions on alternative terms and missing concepts will be enabled and beneficial to the further development of a comprehensive and complete, but not overloaded, RE metamodel.

Acknowledgments. Thanks to Tobias Kaufmann and Klaus Pohl for contributions to the variability model in Section 2. This work is partly funded by VINNOVA within the EASE project.

References

1. Callele, D., Makaroff, D.: Teaching requirements engineering to an unsuspecting audience. In: Proceedings of the 37th SIGCSE technical symposium on Computer science education. pp. 433–437. SIGCSE '06 (2006)
2. Carrillo de Gea, J., Nicolas, J., Aleman, J., Toval, A., Ebert, C., Vizcaino, A.: Requirements engineering tools. Software, IEEE 28(4), 86 –91 (july-aug 2011)
3. Lauesen, S.: Software Requirements - Styles and Techniques. Addison-Wesley (2002)
4. Memon, R.N., Ahmad, R., Salim, S.S.: Problems in requirements engineering education: a survey. In: Proceedings of the 8th International Conference on Frontiers of Information Technology. pp. 5:1–5:6. FIT '10, ACM (2010)
5. Metzger, A., Pohl, K.: Variability management in software product line engineering. In: 29th Int. Conf. on Softw. Eng. pp. 186–187. IEEE (2007)
6. Regev, G., Gause, D.C., Wegmann, A.: Experiential learning approach for requirements engineering education. Requirements Engineering 14(4), 269 – 287 (2009)
7. Regnell, B.: reqt.org – towards a semi-formal, open and scalable requirements modeling tool. In: Requirements Engineering: Foundation for Software Quality (REFSQ), 19th Int. Working Conf. vol. Lecture Notes in Computer Science, 7830, pp. 112–118. Springer (2013)

Appendix A: Definitions of Metamodel Concepts of reqT v3.0

Entity	*Definition*
Actor	A human or machine that communicates with a system.
App	A computer program, or group of programs designed for end users, normally with a graphical user interface. Short for application.
Barrier	Something that makes it difficult to achieve a goal or a higher quality level.
Breakpoint	A point of change. An important aspect of a (non-linear) relation between quality and benefit.
Class	An extensible template for creating objects. A set of objects with certain attributes in common. A category.
Component	A composable part of a system. A reusable, interchangeable system unit or functionality.
Configuration	A specific combination of variants.
Data	Information stored in a system.
Design	A specific realization or high-level implementation description (of a system part).
Domain	The application area of a product with its surrounding entities.
Epic	A large user story or a collection of stories.
Event	Something that can happen in the domain and/or in the system.
Feature	A releasable characteristic of a product. A (high-level, coherent) bundle of requirements.
Function	A description of how input data is mapped to output data. A capability of a system to do something specific.
Goal	An intention of a stakeholder or desired system property.
Idea	A concept or thought (potentially interesting).
Interface	A defined way to interact with a system.
Issue	Something needed to be fixed.
Item	An article in a collection, enumeration, or series.
Label	A descriptive name used to identify something.
Member	An entity that is part of another entity, eg. a field in a in a class.
Meta	A prefix used on a concept to mean beyond or about its own concept, e.g. metadata is data about data.
MockUp	A prototype with limited functionality used to demonstrate a design idea.
Module	A collection of coherent functions and interfaces.
Product	Something offered to a market.
Quality	A distinguishing characteristic or degree of goodness.
Relationship	A specific way that entities are connected.
Release	A specific version of a system offered at a specific time to end users.
Req	Something needed or wanted. An abstract term denoting any type of information relevant to the (specification of) intentions behind system development. Short for requirement.
Resource	A capability of, or support for development.
Risk	Something negative that may happen.
Scenario	A (vivid) description of a (possible future) system usage.
Screen	A design of (a part of) a user interface.
Section	A part of a (requirements) document.
Service	Actions performed by systems and/or humans to provide results to stakeholders.
Stakeholder	Someone with a stake in the system development or usage.
State	A mode or condition of something in the domain and/or in the system. A configuration of data.
Story	A short description of what a user does or needs. Short for user story.
System	A set of interacting software and/or hardware components.
Target	A desired quality level or goal .
Task	A piece of work (that users do, maybe supported by a system).
Term	A word or group of words having a particular meaning.
Test	A procedure to check if requirements are met.
Ticket	(Development) work awaiting to be completed.
UseCase	A list of steps defining interactions between actors and a system to achieve a goal.
User	A human interacting with a system.
Variant	An object or system property that can be chosen from a set of options.
VariationPoint	An opportunity of choice among variants.
WorkPackage	A collection of (development) work tasks.

Attribute	*Definition*
Benefit	A characterisation of a good or helpful result or effect (e.g. of a feature).
Capacity	The largest amount that can be held or contained (e.g. by a resource).
Code	A collection of (textual) computer instructions in some programming language, e.g. Scala. Short for source code.
Comment	A note that explains or discusses some entity.
Constraints	A collection of propositions that restrict the possible values of a set of variables.
Cost	The expenditure of something, such as time or effort, necessary for the implementation of an entity.
Damage	A characterisation of the negative consequences if some entity (e.g. a risk) occurs.
Deprecated	A description of why an entity should be avoided, often because it is superseded by another entity, as indicated by a 'deprecates' relation.
Example	A note that illustrates some entity by a typical instance.
Expectation	The required output of a test in order to be counted as passed.
FileName	The name of a storage of serialized, persistent data.
Frequency	The rate of occurrence of some entity.
Gist	A short and simple description of an entity, e.g. a function or a test.
Image	(The name of) a picture of an entity.
Input	Data consumed by an entity,
Max	The maximum estimated or assigned (relative) value.
Min	The minimum estimated or assigned (relative) value.
Order	The ordinal number of an entity (1st, 2nd, ...).
Output	Data produced by an entity, e.g. a function or a test.
Prio	The level of importance of an entity. Short for priority.
Probability	The likelihood that something (e.g. a risk) occurs.
Profit	The gain or return of some entity, e.g. in monetary terms.
Spec	A (detailed) definition of an entity. Short for specification
Status	A level of refinement of an entity (e.g. a feature) in the development process.
Text	A sequence of words (in natural language).
Title	A general or descriptive heading.
Value	An amount. An estimate of worth.
Why	A description of intention. Rationale.

Relation	*Definition*
binds	Ties a value to an option. A configuration binds a variation point.
deprecates	Makes outdated. An entity deprecates (supersedes) another entity.
excludes	Prevents a combination. An entity excludes another entity.
has	Expresses containment, substructure. An entity contains another entity.
helps	Positive influence. A goal helps to fulfil another goal.
hurts	Negative influence. A goal hinders another goal.
impacts	Some influence. A new feature impacts an existing component.
implements	Realisation of. A module implements a feature.
interactsWith	Communication. A user interacts with an interface.
is	Sub-typing, specialization, part of another, more general entity.
precedes	Temporal ordering. A feature precedes (is implemented before) another feature.
relatesTo	General relation. An entity is related to another entity.
requires	Requested combination. An entity is required (or wished) by another entity.
superOf	Super-typing, generalization, includes another, more specific entity.
verifies	Gives evidence of correctness. A test verifies the implementation of a feature.