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

Björn Regnell

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

Abstract. [**Context & motivation**] This research preview 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 seek to engage computer science students in Requirements Engineering (RE) through an open source requirements engineering 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 that 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 if the respondents is willing to invest around 30 minutes of their time. 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. Some terms can be viewed as "essential RE concepts" in that a many use them and agree on their meaning. The results are encouraging for future work on empirical validation of the relevance of the reqT metamodel.

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

1 Introduction

There are many challenges in teaching Requirements Engineering (RE) [4,5], including conveying 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 our taught metamodel for requirements? This paper investigates this questions in conjunction with the on-going work of developing a metamodel for reqT [8], an open source requirements engineering tool [6] 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 15 Swedish RE scholars are analysed and discussed and conclusions suggest that

a large subset of the concepts of the current reqT metamodel can be argued to be ”essential” in that a majority of the subjects use them while agreeing with the concept definitions. The presented work is an initial validation and further work involving more subjects is needed to provide conclusions with more certainty.

2 Background and Related Work

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 [6] 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 the Computer Science & Engineering program [7].

A critical issue is how to find 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.

Version 3.0 of the metamodel includes the concepts defined in Appendix A. These concepts and definitions are gatherd from various sources including the IREB Glossary [**?**], wikipedia, terminology from agile development, variability modelling terminology, and the Lauesen text book [3] used in the RE course at Lund University [7], in which reqT is applied in student role-playing projects.

!!! EXPLAIN EXAMPLE !!!

```
Model(
  Component("apperance") 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("apperance") requires VariationPoint("shape"),
  App("free") requires Component("apperance"),
  App("free") binds (
    VariationPoint("shape") binds Variant("round")),
  App("premium") requires Component("apperance"),
  App("premium") binds (
    VariationPoint("color") binds (Variant("red"), Variant("green")),
    VariationPoint("shape") binds (Variant("round"), Variant("square")),
    VariationPoint("payment") binds Variant("cash")))
```

3 Methodology and Data Collection

!!! explain how sources such as the IREB glossary was used to form the metamodel concept definitions !!!

4 Data Analysis

Subject background. The background questions in the survey regards the role of the subject, as shown in Table 1. In summary, the included[1] total number of subjects is 14, of which 10 are teachers, 10 are developers and 13 are researchers.

Table 1. Background of subjects, $N = 15$. The table shows anonymized subject ids

Question	*Subject responding YES*
Do you teach software engineering and/or requirements engineering? YES/NO	R01 R03 R04 R05 R07 R08 R09 R11 R12 R14
Do you develop software by writing code and/or creating system models? YES/NO	R01 R02 R03 R07 R08 R09 R10 R11 R13 R14
Do you do academic research in software and/or requirements engineering? YES/NO	R01 R03 R04 R05 R06 R07 R08 R09 R10 R11 R12 R13 R14

Frequency analysis. The idea of ”essentiality” is characterized as the number of subjects that has responded that they (1) use the concepts at least in an informal, non-persistent way, and (2) use the concepts in a similar meaning as in the definition in Appendix A. Table 2 shows the results of the frequency counts.

[1] One subject answered NO on all background questions and was therefore excluded.

Table 2. Frequency analysis of concepts' "essentiality", where n is the number of subjects that answered $(use >= 1)$ and $(agree = 2)$ for each concept.

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`
5			
4	`Ticket`		
3			
2			
1	`Epic`	`Gist`	
0			

5 Discussion and Conclusion

!!Discuss and conclude!!

Limitations. Due to the limited number of subjects and the high degree of homogeneity among subjects with respect to background, it is difficult to analyse and draw conclusions about potential differences in opinions between e.g. teachers and developers. Some subjects needed more time and completed survey offline during the coming days, which may give a variation in how careful the responses were considered. The survey was conducted by the author and inventor of reqT, in conjunction with a seminar and demo of reqT. In order to avoid any positive bias due to advocacy in favour of the reqT metamodel, the survey was held prior to the seminar and demo. This in turn may introduce a threat of limited knowledge among subjects of the idea behind the modelling approach in reqT

Future work.

Acknowledgments. 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. Regev, G., Gause, D.C., Wegmann, A.: Experiential learning approach for requirements engineering education. Requirements Engineering 14(4), 269 – 287 (2009)
6. Regnell, B.: reqt.org – towards a semi-formal, open and scalable requirements modeling tool. In: Requirements Engineering: Foundation for Software Quality (REFSQ), 19th International Working Conference. vol. Lecture Notes in Computer Science, 7830, pp. 112–118. Springer (2013)
7. Requirements Engineering course home page at Lund University: `http://cs.lth.se/ets170`, visited Oct 2015.
8. The reqT open source tool home page: `http://reqT.org`, visited Oct 2015.

Appendix A: Definitions of Metamodel Concepts

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.