# What is essential?

## – A pilot survey on views about the requirements metamodel of reqT.org

Björn Regnell

Lund University, Sweden

refsq.org/2016
github.com/bjornregnell/reqT-survey

1 Objective
- Research Question
- Approach

2 Background
- About reqT

3 Methodology
- Data Collection
- Data Analysis

4 Results
- Essentiality

5 Conclusion
- Contribution & Future Work

# Research question

In the context of software requirements engineering education:

- How to choose a set of **essential requirements engineering concepts** that allows for **sufficient expressiveness**, without overloading the metamodel with esoteric concepts just for the sake of **completeness**?

# Research question

In the context of software requirements engineering education:

- How to choose a set of **essential requirements engineering concepts** that allows for **sufficient expressiveness**, without overloading the metamodel with esoteric concepts just for the sake of **completeness**?
- Presumption: as teachers we should be method agnostic; there is no single ”correct” dogma

# Approach

- Make a survey among RE scholars

# Approach

- Make a survey among RE scholars
  - How to quantify ”essentiality”?

# Approach

- Make a survey among RE scholars
  - How to quantify ”essentiality”?
    - One possible quantification:
      The more scholars that **agree** on a definition of a concept
      and
      the more scholars that **use** the concept,
      the more **essential** is the concept.

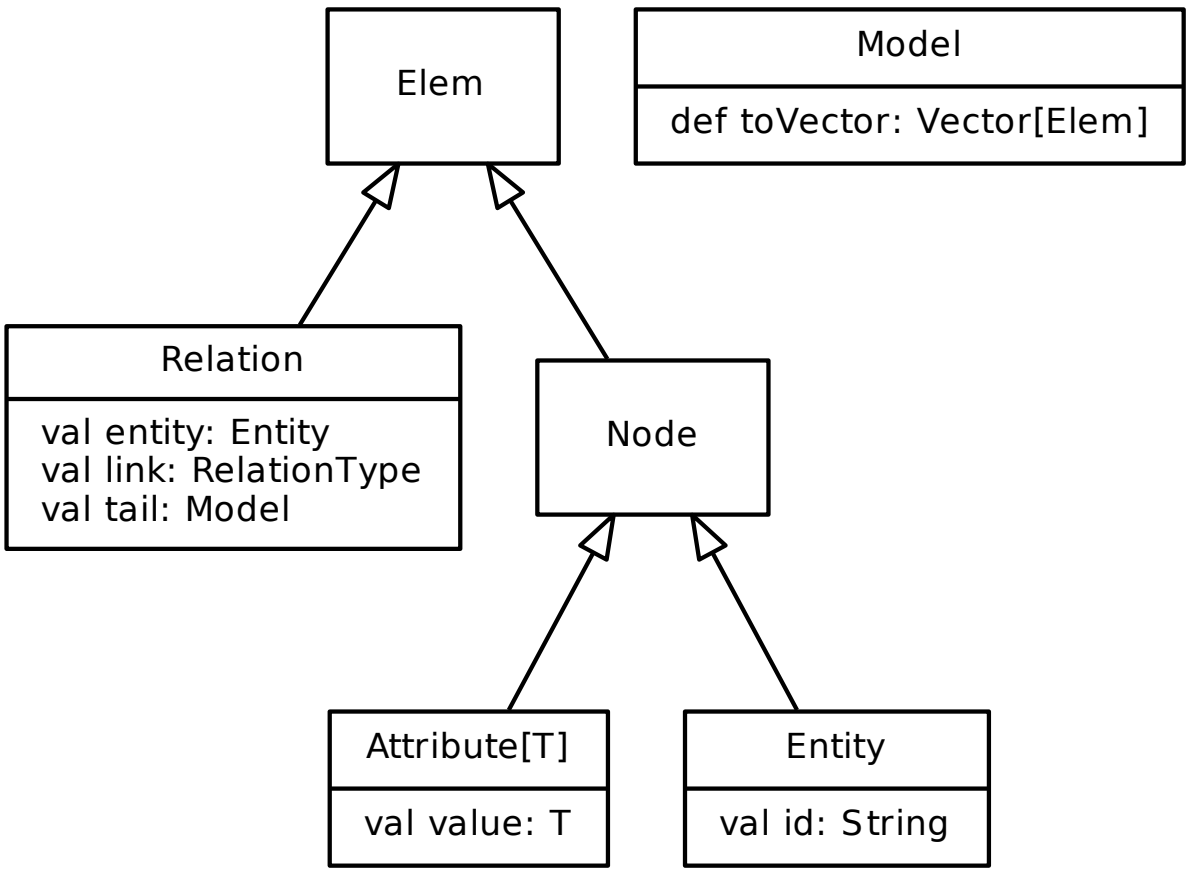
# Approach

- Make a survey among RE scholars
  - How to quantify ”essentiality”?
    - One possible quantification:
      The more scholars that **agree** on a definition of a concept
      and
      the more scholars that **use** the concept,
      the more **essential** is the concept.
- Use the reqT.org metamodel as a basis for the survey

# What is reqT?

- reqT is an open source requirements engineering tool developed in an **educational** context
  Download from here: `http://reqt.org/`
  `java -jar reqT.jar`
- reqT aims to inspire **code-loving** cs students to learn more about RE through a **Scala**-internal DSL
- has a metamodel with **entities**, **relations** and **attributes** gathered from various sources such as
  text books, research papers, IREB, wikipedia, etc.

# The base classes of the reqT metamodel



# A small reqT model

```
val m = Model(
  Feature("chat") has (
    Spec("The system shall enable sending text messages among users"),
    Prio(42)
  )
)
```

# An larger example reqT model

```
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")))
```

# 49 entities of the metamodel (see paper appendix)

| *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. |

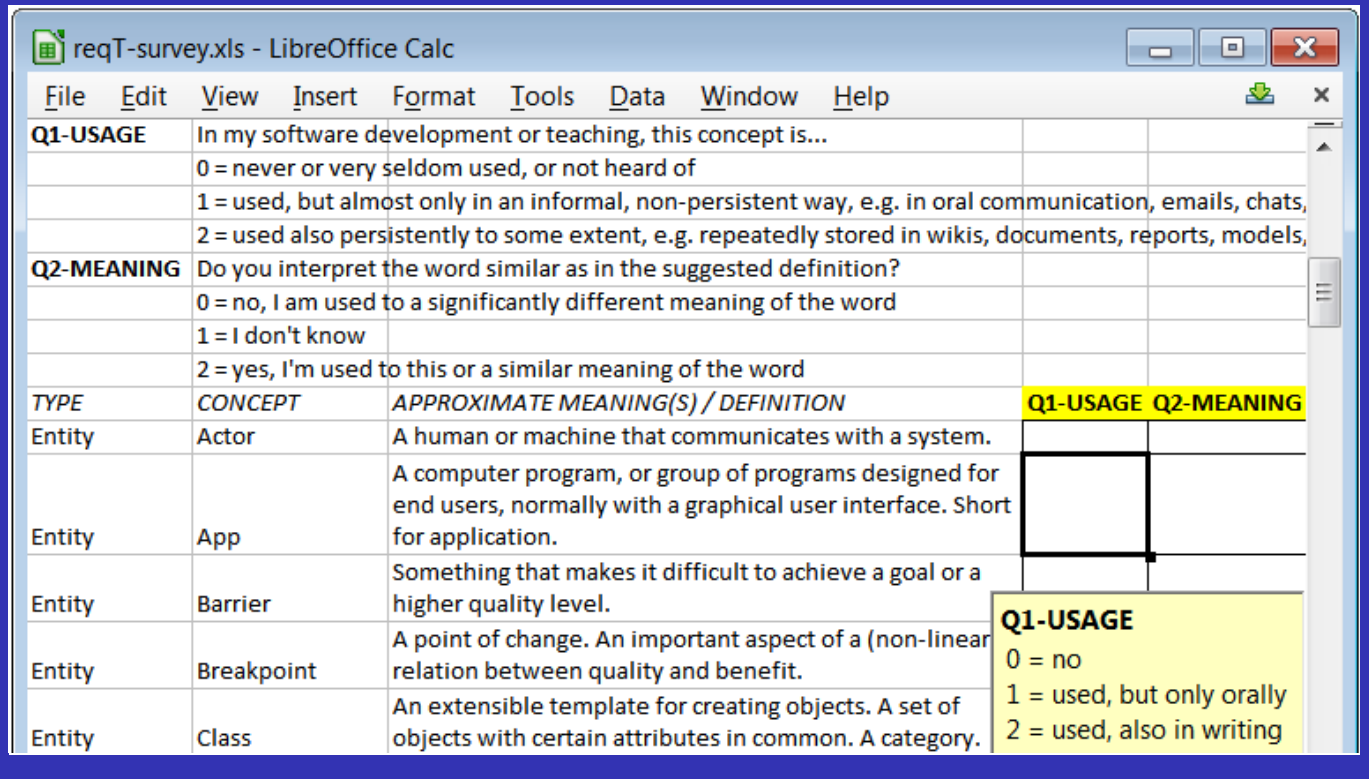
# 28 attributes of the metamodel (see paper appendix)

| ***Attributes*** | ***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 |
| `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 indic |
| `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. |

# 15 realtions of the metamodel (see paper appendix)

| ***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 fe |
| `relatesTo` | General relation. An entity is related to another entity. |
| `requires` | Requested combination. An entity is required (or wished) by another entit |
| `superOf` | Super-typing, generalization, includes another, more specific entity. |
| `verifies` | Gives evidence of correctness. A test verifies the implementation of a fea |

# Data Collection



reqT-survey.xls - LibreOffice Calc

File Edit View Insert Format Tools Data Window Help

| | | | | |
|---|---|---|---|---|
| **Q1-USAGE** | In my software development or teaching, this concept is... | | | |
| | 0 = never or very seldom used, or not heard of | | | |
| | 1 = used, but almost only in an informal, non-persistent way, e.g. in oral communication, emails, chats, | | | |
| | 2 = used also persistently to some extent, e.g. repeatedly stored in wikis, documents, reports, models, | | | |
| **Q2-MEANING** | Do you interpret the word similar as in the suggested definition? | | | |
| | 0 = no, I am used to a significantly different meaning of the word | | | |
| | 1 = I don't know | | | |
| | 2 = yes, I'm used to this or a similar meaning of the word | | | |
| *TYPE* | *CONCEPT* | *APPROXIMATE MEANING(S) / DEFINITION* | **Q1-USAGE** | **Q2-MEANING** |
| Entity | Actor | A human or machine that communicates with a system. | | |
| Entity | App | A computer program, or group of programs designed for end users, normally with a graphical user interface. Short for application. | | |
| Entity | Barrier | Something that makes it difficult to achieve a goal or a higher quality level. | | |
| Entity | Breakpoint | A point of change. An important aspect of a (non-linear relation between quality and benefit. | | |
| Entity | Class | An extensible template for creating objects. A set of objects with certain attributes in common. A category. | | |

**Q1-USAGE**
0 = no
1 = used, but only orally
2 = used, also in writing

Q1 use = {no | used, but only orally | used, also in writing}
Q2 agree = {no, different meaning | don't know | yes, similar meaning}
Answered by 15 swedish RE scholars (100% response rate)
https://github.com/bjornregnell/reqT-survey

# Data Analysis

**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.

# Frequency Analysis

| *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 | |

# Essentiality Definition

- If an ”essentiality threshold” is chosen at $N/2$ then only the 9 concepts from row $n = 7$ and below are considered ”non-essential”
- **More than 90% of the 92 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.

# Omissions

**Missing Concepts** according to subjects:
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.

# Conlusion

- Contribution
  - A way to quantify ”essentiality” – tested in a pilot survey
  - A set of potentially ”essential” concepts (according to 14 RE scolars in Sweden)
- Future Work
  - Increase validity
  - Investigate omissions