Why Are Ontologies and Languages for Software Quality Increasingly Important?

Xavier Franch, Full Professor, Polytechnic University of Catalonia (BarcelonaTech) – franch@essi.upc.edu

December 11 | 1:00 PM ET

- Today’s session will be recorded.
- An archive of today’s talk will be available at: www.sercuarc.org/serc-talks/
- Use the Q&A box to queue questions, reserving the chat box for comments, and questions will be answered during the last 5-10 minutes of the session.
- If you are connected via the dial-in information only, please email questions or comments to SERCtalks@stevens.edu.
- Any issues? Use the chat feature for any technical difficulties or other comments, or email SERCtalks@stevens.edu.
The Systems Engineering Research Center (SERC) is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology.

Any views, opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense, ASD(R&E), nor the SERC.

No Warranty. This Stevens Institute of Technology Material is furnished on an “as-is” basis. Stevens Institute of Technology makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Stevens Institute of Technology does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

This material has been approved for public release and unlimited distribution.
Why are Ontologies and Languages for Software Quality Increasingly Important?

Prof. Xavier Franch
UPC-BarcelonaTech, Barcelona, Catalonia (Spain)
SERC Talk, December 2018
Outline

- Motivation
- Ontologies
- Quality models
  - How are they used in practice
- Customization of quality models
  - An example: the Q-Rapids project
  - Another example: requirement patterns
- Expressing ontologies
  - An example: the NFR framework
- Summary
Motivation

Why are Ontologies and Languages for Software Quality Increasingly Important?

Problem — Customers struggle to state functional requirements. They do not understand non-functional requirements.

“...a failure to satisfy a non-functional requirement can be critical, even catastrophic...”

Bill Curtis, SERC Talk Oct. 2018
Quality is elusive...

- Difficult to specify
- Difficult to quantify
- Difficult to verify
- **Difficult to agree upon**
“A property, or quality, that the product must have, such as an appearance, or a speed or accuracy”

“Describe the nonbehavioral aspects of a system, capturing the properties and constraints under which a system must operate”

“Requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors”

“A requirement that specifies physical constraints on a functional requirement”

“They place restrictions on the product being developed and the development process, and they specify external constraints that the product must meet”
Example: categories of quality

“A property, or quality, that the product must have, such as an **appearance**, or a **speed** or **accuracy**”

“The required overall attributes of the system, including **portability**, **reliability**, **efficiency**, **human engineering**, **testability**, **understandability**, and **modifiability”**

“A requirement that specifies system properties, such as **environmental** and implementation constraints, **performance**, platform dependencies, **maintainability**, **extensibility**, and **reliability”**

“Global requirements on its **development** or **operational cost**, **performance**, **reliability**, **maintainability**, **portability**, **robustness**, and the like”
Categories of quality aspects
How to cope with this diversity?

Concept of **ontology**

- Branch of philosophy which deals specially with the nature of being
- Explicit specification of a shared conceptualization
- Provide a precise and unambiguous terminology
Use of ontologies in RE

Three main categories:
Use of ontologies in RE

Three main categories:

• Structure of specification documents
Use of ontologies in RE

Three main categories:

• Structure of specification documents
• Fundamental ontologies
Use of ontologies in RE

Three main categories:

• Structure of specification documents
• Fundamental ontologies
• Application domain ontologies
Expressing an ontology

An ontology needs to represent:

• Lexicon of concepts
• Taxonomical arrangement
• Properties among concepts
Some approaches exist for quality

An Initial Ontology for System Qualities

Barry Boehm, Nupul Kukreja

First published: 29 October 2015  |  https://doi.org/10.1002/j.2334-5837.2015.00067.x

The primary need for a system qualities ontology is to enable more effective collaboration in system definition, development, and evolution across system stakeholders from different disciplines.
Quality Models (QM)

- A lightweight implementation of ontologies for quality:
  
  “The set of characteristics and the relationships between them which provide the basis for specifying quality requirements and evaluating quality”

- Characteristics ≅ quality attributes
- Relationships ≅ (taxonomic) decomposition
Example: ISO/IEC 25010 – Terms

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance efficiency</td>
<td>performance relative to the amount of resources used under stated conditions</td>
</tr>
<tr>
<td>Compatibility</td>
<td>degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions, while sharing the same hardware or software environment</td>
</tr>
<tr>
<td>Usability</td>
<td>degree to which a product or system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use</td>
</tr>
<tr>
<td>Reliability</td>
<td>degree to which a system, product or component performs specified functions under specified conditions for a specified period of time</td>
</tr>
<tr>
<td>Security</td>
<td>degree to which a product or system protects information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization</td>
</tr>
<tr>
<td>Maintainability</td>
<td>degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers</td>
</tr>
<tr>
<td>Portability</td>
<td>degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational or usage environment to another</td>
</tr>
</tbody>
</table>
ISO/IEC 25010 - Metamodel

- Quality Model
  - Quality Feature
    - Characteristic
      - Subcharacteristic
        - Derived Subcharacteristic
        - Basic Subcharacteristic
      - Derived Subcharacteristic
      - Basic Subcharacteristic
    - Subjective
    - Objective
  - Derived Attribute
  - Basic Attribute
  - Metric
  - {disjoint, complete}
How does the community work?
Based on a literature review

65 works presenting 47 different proposals of quality models in the period 2001-2013
Results

56% of the approaches are new proposals

Variable structure
Use in practice

More than 85% of companies may need customized quality models, adapted to their needs.

We need effective ways to build such customized quality models while enforcing ontological understanding as much as possible.
Customization

Domain

Company C1

Domain QM

Company C2

C1’s QM

C2’s QM
Building a domain model

- Locate the appropriate sources
- Organize the existing knowledge
  - See Web Services case
- Analyse the quality (sub)characteristics that are more relevant in the domain
  - E.g., using the ISO/IEC 25010 quality model (or any other)
- Identify quality attributes
  - Effort in those (sub)characteristics that are more relevant
## WS case: ISO/IEC 25010 coverage

<table>
<thead>
<tr>
<th></th>
<th>Functional suitability</th>
<th>Performance efficiency</th>
<th>Compatibility</th>
<th>Usability</th>
<th>Reliability</th>
<th>Security</th>
<th>Maintainability</th>
<th>Portability</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Y+</strong></td>
<td>2%</td>
<td>49%</td>
<td>0%</td>
<td>6%</td>
<td><strong>77%</strong></td>
<td>55%</td>
<td>11%</td>
<td>2%</td>
</tr>
<tr>
<td><strong>Y</strong></td>
<td>2%</td>
<td>19%</td>
<td>0%</td>
<td>4%</td>
<td>4%</td>
<td>19%</td>
<td>6%</td>
<td>2%</td>
</tr>
<tr>
<td><strong>P+</strong></td>
<td>6%</td>
<td>17%</td>
<td>2%</td>
<td>4%</td>
<td>11%</td>
<td>4%</td>
<td>4%</td>
<td>2%</td>
</tr>
<tr>
<td><strong>P</strong></td>
<td>57%</td>
<td>6%</td>
<td>34%</td>
<td>13%</td>
<td>8%</td>
<td>4%</td>
<td>15%</td>
<td>40%</td>
</tr>
<tr>
<td><strong>ND</strong></td>
<td>32%</td>
<td>9%</td>
<td>64%</td>
<td><strong>72%</strong></td>
<td>0%</td>
<td>17%</td>
<td>64%</td>
<td>53%</td>
</tr>
</tbody>
</table>
### WS case: ISO/IEC 25010 coverage

#### 5. Reliability

<table>
<thead>
<tr>
<th></th>
<th>5.1 Maturity</th>
<th>5.2 Availability</th>
<th>5.3 Fault tolerance</th>
<th>5.4 Recoverability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y+</td>
<td>0%</td>
<td>28%</td>
<td>13%</td>
<td>23%</td>
</tr>
<tr>
<td>Y</td>
<td>2%</td>
<td>66%</td>
<td>26%</td>
<td>6%</td>
</tr>
<tr>
<td>P+</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>2%</td>
</tr>
<tr>
<td>P</td>
<td>11%</td>
<td>0%</td>
<td>0%</td>
<td>13%</td>
</tr>
<tr>
<td>(ND)</td>
<td>0%</td>
<td>87%</td>
<td>6%</td>
<td>55%</td>
</tr>
</tbody>
</table>

#### 4. Usability

<table>
<thead>
<tr>
<th></th>
<th>4.1 Appropriate recognizability</th>
<th>4.2 Learnability</th>
<th>4.3 Operability</th>
<th>4.4 User error protection</th>
<th>4.5 User interface aesthetics</th>
<th>4.6 Accessibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y+</td>
<td>6%</td>
<td>0%</td>
<td>4%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Y</td>
<td>4%</td>
<td>4%</td>
<td>2%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>P+</td>
<td>4%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>P</td>
<td>13%</td>
<td>6%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>(ND)</td>
<td>72%</td>
<td>85%</td>
<td>95%</td>
<td>91%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>
## Defining quality attributes

<table>
<thead>
<tr>
<th>ISO/IEC 25010 subcharacteristics</th>
<th>Quality attribute</th>
<th>Definition</th>
<th>Definition usage %</th>
<th>Name usage %</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Functional suitability</strong></td>
<td><strong>Accuracy</strong></td>
<td>Error rate produced by the service [26][28].</td>
<td>62%</td>
<td>42%</td>
</tr>
<tr>
<td><strong>Performance efficiency</strong></td>
<td><strong>Accessibility</strong></td>
<td>Degree the service is capable of serving a web service request [53][16].</td>
<td>47%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Capacity</strong></td>
<td><strong>Capacity</strong></td>
<td>Limit of concurrent requests for guaranteed performance [26][55].</td>
<td>51%</td>
<td>65%</td>
</tr>
<tr>
<td><strong>Time behaviour</strong></td>
<td><strong>Response Time</strong></td>
<td>Time to complete a WS request (from a client perspective) [26][28].</td>
<td>83%</td>
<td>73%</td>
</tr>
<tr>
<td><strong>Time behaviour</strong></td>
<td><strong>Throughput</strong></td>
<td>Number of web service requests served at a given time period [28][16].</td>
<td>60%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Compatibility</strong></td>
<td><strong>Interoperability</strong></td>
<td>Ease with which a consumer application or agent interoperates with a service [27].</td>
<td>36%</td>
<td>100%</td>
</tr>
<tr>
<td><strong>Reliability</strong></td>
<td><strong>Availability</strong></td>
<td>Probability that the service can respond to consumer requests [51][27].</td>
<td>94%</td>
<td>98%</td>
</tr>
<tr>
<td><strong>Recoverability</strong></td>
<td><strong>MTTR</strong></td>
<td>Mean Time to Repair [30][27].</td>
<td>30%</td>
<td>8%</td>
</tr>
<tr>
<td><strong>Fault tolerance</strong></td>
<td><strong>Robustness</strong></td>
<td>Degree to which a service can function correctly in the presence of invalid, incomplete or conflicting inputs [26][28].</td>
<td>38%</td>
<td>94%</td>
</tr>
<tr>
<td><strong>Security</strong></td>
<td><strong>Authentication</strong></td>
<td>Measure of how the service authenticates principals (users or other services) who can access service and data [26].</td>
<td>45%</td>
<td>95%</td>
</tr>
<tr>
<td><strong>Confidentiality</strong></td>
<td><strong>Confidentiality</strong></td>
<td>Measure of how the service treats the data, so that only authorized principals can access or modify the data [26].</td>
<td>45%</td>
<td>85%</td>
</tr>
</tbody>
</table>
Customizing a domain QM

Operationalised product quality models and assessment: The Quamoco approach

Stefan Wagner, Andreas Goeb, Lars Heinemann, Michael Kläs, Constanza Lampasona, Klaus Lochmann, Alois Mayr, Reinhold Plösch, Andreas Seidl, Jonathan Streit, Adam Trendowicz

- Use their own metamodel
- Focus on internal quality
- Emphasis on modularity

I present some simplification/customization of the method
1. Definition of metrics

- In the basis of:
  - which are the business priorities of the project
  - what data is available (i.e., can be effectively used)
1. Definition of metrics

- **Attribute**
- **Business value**

- **Normalized metric**

- **Raw metric**
- **Size metric**
- **Raw metric**

- **Data source**
- **Data availability**
- **Data source**
2. Evaluation of attributes

- Determination metrics’ impact onto attributes
  - Use of (linear) utility functions
    - Utility: strength of preference a decision-maker has among alternatives regarding a specific quality aspect

```
<table>
<thead>
<tr>
<th>Raw metric</th>
<th>Attribute</th>
</tr>
</thead>
</table>
```

Thresholds coming from benchmarking
3. Aggregation of factors

- Synthesis of individual utilities into global one
  - Weights for contributing factors

Weights coming from existing data or expert opinion
4. Interpretation

- Translation of utility values into a QA value
  - Interpretation model transforming from ratio to ordinal
  - Could be done at the level of general quality, a characteristic or a subcharacteristic
  - E.g., analogy with school grade
Example: The Q-Rapids project

- Candidate / improved QRs
- QR-related key indicators
- Decision making
- QRs scheduled
- Rapid development process
- Running System
- Repositories
- Mined data
- Data gathering
- Data Analysis
- Strategic Analysis
Example: The Q-Rapids project

Decision maker → Product Backlog → Development process → End-user

Decision rules
Constraints

Requirements assessment

Repositories (software, project)

Runtime data (usage, QoS)

Strategic dashboard

Mined data

International Working Conference on Requirements Engineering: Foundation for Software Quality
REFSQ 2017: Requirements Engineering: Foundation for Software Quality pp 167-173 | Cite as


Authors

Liliana Guzmán, Marc Oriol, Pilar Rodriguez, Xavier Franch, Andreas Jedlitschka, Markku Olivo
Overall view

Legend
- Quality Aspect
- Product/Process Factor
- Assessed Metric
- Raw Data
- Data Source

Maintainability Value = 0.26

Code Quality

AM1: Commented files
$U_1(M1) = 0.5$

AM2: Non-complex files

Absence of duplications

$U_2(M3/M4) = 0.4$

M1: Density of comments of a file
M2: Total number of files
M3: Cyclomatic complexity of a file
M4: Number of functions of a file

Static Sw. Code Analysis from SonarQube

A Bottom-Up Approach

IEEE Xplore
Digital Library

Institutional Sign In

Conferences > 2018 44th Euromicro Conference...

A Quality Model for Actionable Analytics in Rapid Software Development

4 Author(s) Silverio Martínez-Fernández; Andreas Jedlitschka; Liliana Guzmán; Anna Maria Vollmer View All Authors

IEEE
Complex ecosystem

Software Development Process

ISSUE: epics, use cases, users stories, subtasks
With EFFORT: story points

Jira

sprint planning

PO

ITERATION: a.k.a. Sprint

Jira

CODE REVIEW

Gerrit

RESOURCES: employees’ experience with technologies

Proprietary tool

Software System

DOCUMENTATION:
gerated from code, in the development environment

SVN | Confluence

COMMIT

Gerrit | Plugin to link with issue id.

Team

accepts commit

SOURCE CODE: main line

git
decides when to release

PO

RELEASE: code, issues, date

git | Jira

BUILD, TEST: automatic in development environment

Jenkins

BUILD, TEST: stress and stability testing

Jenkins

STATIC CODE ANALYSIS:
quality metrics, quality rule violations

CodeSonar | SonarQube

BUILD, TEST: automatic in development environment

Jenkins

SOFTWARE USERS

CUSTOMER FEEDBACK

ServiceDesk | Digium Enterprise

LOGS: crashes, errors, bugs, performance results, usage statistics

Elastic | SafeMove Analytics

Legend

ENTITY: details and characteristics

Data source/tool

Blue → Sw. Development Process

Red → Sw. System

Green → Sw. Users

Black → Actions

39
Data Integration ontology
Example of application
Reuse adoption in industry

- 1 - Inexistent or very low: 6%
- 2 - Low: 21%
- 3 - Medium: 21%
- 4 - High: 32%
- 5 - Very high: 6%

Organizations tried it, but did not succeed
- 2 (13.55%)
- 3 (20.00%)

Organizations do not think it would bring any different in the long run
- 6 (40.00%)

Organizations' projects are too different one from each other
- 7 (46.67%)

Organizations never thought about it
- 7 (46.67%)

Organizations consider it too complex
- 8 (53.33%)

Even if it may provide benefit, the initial investment is too high
- 14 (93.33%)

Organizations do not know how to do it
Approaches to reuse

- Copy & Paste of Individual reqs.: 27 (69.77%)
- Copy & Paste of Groups of reqs.: 28 (65.11%)
- Duplicate of a full reqs. specification: 25 (58.14%)
- Fill in predefined templates: 17 (39.53%)
- Other: 6 (13.95%)
- Varies depending on the project: 15 (34.88%)
- Use of a req. patterns catalogue: 1 (2.32%)

Total: 85
Pattern approach to reuse
Requirement patterns

SRP: Supplier Economic Information
Goal: Assessing the economic situation of the supplier

Does the Customer require specific conditions on the economic situation of the supplier?

Yes

Economic Situation Prerequisites Form

SRP Form

- Fixed: The supplier shall fulfill some economic situation prerequisites.
- Extended 1: The supplier’s company shall have a minimum net income of amount currencyUnit on the last amountOfTime timeUnit.
- Extended 2: The supplier’s company shall have a minimum turnover of amount currencyUnit on the last amountOfTime timeUnit.

Constraints

- Fixed part cannot be applied more than once.
- Extended 1 and Extended 2 parts cannot be applied more than once.

No

Economic Situation Information Form

SRP Form

- Fixed: The supplier shall provide economic information of its company.
- Extended 1: The supplier shall provide information of its company’s turnover on the last amountOfTime timeUnit.
- Extended 2: The supplier shall provide information of its company’s net income on the last amountOfTime timeUnit.

SRP Parts

- Fixed part cannot be applied more than once.
- Extended 1 and Extended 2 parts cannot be applied more than once.

Parameter Metrics

- amount, amountOfTime: integer (inv integer must be greater than 0)
- timeUnit: domain [years, months, ...]
- currencyUnit: domain [USD, EUR, GBP, JPY, ...]
Use of the taxonomy
Expressing an ontology

Different formalisms have been used:

- OWL
- SPARQL
- RDF
- DL
- SWRL
- ad hoc
- FOL
- UML
- XML
- ER
Example *ad hoc*: The NFR framework

The NFR framework was delivered in year 2000 after some efforts starting in early nineties:

- created by J. Mylopoulos and team
- emerged in parallel to the $i^*$ framework (Yu 1995)
- has become the most popular approach to NFR modeling and reasoning
Main characteristics

- acquiring or accessing knowledge about the domain
  - functional requirements, priorities, ...
- identifying particular NFRs for the domain
  - use of NFR type catalogues
- decomposing NFRs
- identifying feasible operationalizations, and selecting them
- dealing with ambiguities, conflicts and dependencies
- support decisions with design rationale
- evaluating the impact of decisions
The basis: goal decomposition

- Collect timetables
  - By Person
  - By System
    - Collect from agents
    - Collect from users
      - Send request
      - Receive request
      - Manually
      - Automatically
    - Choose schedule
      - Schedule meeting
        - AND
          - AND
Keypoint: notion of softgoal

- NFR Softgoal: high-level, non-operationalized, NFR
  - Softgoal can rarely be completely satisfied, hence goal is regarded satisfied (or satisfice) within acceptable limits

- Operationalizing Softgoal: possible solutions or design alternatives which help achieving the NFRs

- Claim Softgoal: justify the rationale and explain the context for a softgoal or interdependency link

- Softgoal consist of:
  - NFR Type (e.g. Security; Authentication)
  - One or more topic to indicate meaning and information item (e.g. [CardData]; [Account]).
Interdependency graph

- Interdependencies: refinement of softgoals and the contributions of offspring softgoals towards the achievement of its parent
- Softgoal Interdependency Graph (SIG): graph where softgoals and their interdependencies are represented
Language

NFR softgoal  Claim softgoal  Operationalizing softgoal

\[ \begin{align*}
\text{BREAK} & \quad \text{HURT} & \quad \text{UNKNOWN} \\
\text{SOME}^- & \quad \text{HELP} & \quad \text{MAKE} \\
\text{SOME}^+ & \quad \text{EQUAL} & \quad \text{AND} \\
\text{OR} & \quad \text{contribution} \\
\end{align*} \]

\[ \begin{align*}
\text{x} & \quad \text{denied} \\
\text{w}^- & \quad \text{weakly denied} \\
\text{u} & \quad \text{undecided} \\
\text{w}^+ & \quad \text{weakly satisficed} \\
\checkmark & \quad \text{satisficed} \\
\end{align*} \]
Example of model

- Space for accounts
  - Use uncompressed formats
    - Claim “Optimized validation will not hurt response time much”

- Response time for accounts
  - Use indexing techniques
  - Validate access against eligibility rules

- Accurate accounts
  - Claim “Accuracy has been the first wish of most stakeholders”
  - Authorize access to account information

- Identify users
  - ...
Evaluation procedure (excerpt)

(Chung et al. 2000)
Summary
UPCOMING TALKS:
2019

Themes:
February, April, June 2019:
Continuous Delivery and Deployment of Cyber-Physical-Human Systems

August, October, December 2019:
Dependable Autonomy

Please visit the SERC Talks page for more information and updates.
Thank you for joining us!

Please check back on the [SERC website](#) for today’s recording and future SERC Talks information!

Subscribe and follow SERC on these channels:

CONTACT

Editor-in-Chief: Dr. Barry Boehm, University of Southern California – [boehm@usc.edu](mailto:boehm@usc.edu)

Webinar Coordinator: Ms. Mimi Marcus, Stevens Institute of Technology – [SERCtalks@stevens.edu](mailto:SERCtalks@stevens.edu)