Rethinking Functional Requirements: Novel Approaches to Categorizing Requirements

The conceptual power of mathematical modelling

Manfred Broy

Technische Universität München
Institut für Informatik
D-80290 Munich, Germany
Diversity of requirements: A slide due to Michael Jackson

- System Complies with All Safety Regulations!
- Lift Is Easy to Use!
- Lift Comes when I Request and Goes to the Floor I Want!
- Special Mode for Firefighter Operation!
- Efficiency Means Fewer Lifts, More Rentable Space!
- Helpful Operational Procedures for Monthly Maintenance!
- Graceful Service Degradation on Minor Failures!
- No Lower Classes Allowed on the Tycoon Floor!
- Service I can Define to Meet the Varying Usage Demands!
There are many stakeholders with diverse needs, often vaguely understood.
**Air Bag**

Beispiel: **BMW X5 5-Sterne im Euro-NCAP** (Maximalbewertung)

**Crash Euro-NCAP**
- 64km/h
- Deformierbare Barriere
- 50% Offset

Permanent werden alle Daten der Sensorik (1, 2, 4, 8) im Steuergerät (3) ausgewertet.

0 ms

Bereits die ersten cm einer Deformation erzeugen eine Beschleunigung, die ausreicht einen Crash zu erkennen. In der ca. 12ten Millisekunde (0,12 sec) ist die Entscheidung im Steuergerät getroffen die Airbags zu aktivieren.

40 ms

Die Airbags beginnen bereits in der Vorwärtsverlagerung der Passagiere sich zu entfalten.

60 ms

Nun ist die Schutzwirkung in voller Aktion.

80 ms

Die hohe Dynamik des Crashes zeigt sich in der weiteren Bewegung des Fahrzeugs.
Nun wird:
- Zentralverriegelung geöffnet
- Warnblinkanlage aktiviert
- Telefonnotruf abgesetzt.

400 ms
Example: A Short Example: airbag

1. The airbag has to be activated after the crash sensor indicates a crash within the interval of 160 to 180 millisecond. ◊ a problem domain requirement,

2. The probability that the airbag fails to fulfil specification (1) is less than 0.01 %. ◊ a reliability/safety requirement,

3. The probability that the airbag is activated with 170 ± 5 millisecond after the crash sensor indicates a crash is above 99.9 %. ◊ a performance/safety requirement,

4. The probability that the airbag gets activated without a signal from the crash sensor is below 10^{-8} %. ◊ a probabilistic safety property - a problem domain requirement.

5. The sidebag airbag is ordered as an extra by more than 80 % of the customers. ◊ a business requirement
Motivation

• A clear understanding of the notion of “functional” (and “non-functional”) requirements

• Categorizing requirements as a basis for structuring requirements
  ◊ Based on system models
  ◊ Based on quality models
  ◊ Based on context structuring

• Understanding the relationship between different categories of requirements

• Having a formal basis that – at least in principle – supports the formulation of specifications about time and probability
Nonfunctional requirement

- **Wikipedia**
  - a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors

- **University of Toronto**
  - Non-functional requirements are global constraints on a software system

- **http://www.requirementsauthority.com**
  - A Non-Functional Requirement is usually some form of constraint or restriction that must be considered when designing the solution.

- **IEEE**
  - non functional requirement: a software requirement that describes not what the software will do, but how the software will do
Some statements due to Martin Glinz

• Glinz writes "*Although the term non-functional requirements has been in use for more than twenty years, there is no consensus in the requirements engineering community what non-functional requirements are and how we should elicit, document and validate them*."

• However, in the end his particular definition is equally imprecise and fuzzy.
  ◊ in his definition a non-functional requirement is an "attribute of" or a "constraint on" a system.
  ◊ very unsatisfactory as a definition, because it reduces the fuzzy term "non-functional" requirement to the equally fuzzy and general terms such as "attribute" or "constraint".
  ◊ not obvious why system functionality and functional requirements should not also be described by attributes or constraints.
A characterization due to Martin Glinz

M. Glinz:
On Non-Functional Requirements.
In: 15th IEEE International Requirements Engineering Conference (RE '07), 2007

Figure 2. A concern-based taxonomy of requirements
Basic System Notion: What is a discrete system (model)

A system has

- a system **boundary** that determines
  - what is part of the systems and
  - what lies outside (called its context)
- an **interface** (determined by the system boundary), which determines,
  - what ways of interaction (actions) between the system und its context are possible (static or **syntactic interface**)
  - which behavior the system shows from the point of view of the context (**interface behavior**, dynamic interface, interaction view)
- a structure and distribution with an internal structure, given
  - by its structuring in sub-systems (**sub-system architecture**)
  - by its states und state transitions (**state view**, state machines)
- **quality** profile
- the views use a **data model**
- the views may be documented by adequate models
Sets of typed channels

\[ I = \{x_1 : T_1, x_2 : T_2, \ldots \} \]
\[ O = \{y_1 : T'_1, y_2 : T'_2, \ldots \} \]

syntactic interface

\[ (I \rightharpoonup O) \]

data stream of type \( T \)

\[ \text{STREAM}[T] = \{\text{IN}\setminus\{0\} \rightarrow T^*\} \]

valuation of channel set \( C \)

\[ \text{IH}[C] = \{C \rightarrow \text{STREAM}[T]\} \]

interface behaviour for syn. interface \((I \rightharpoonup O)\)

\[ [I \rightharpoonup O] = \{\text{IH}[I] \rightarrow \emptyset(\text{IH}[O])\} \]

interface specification

\[ p: I \cup O \rightarrow IB \]

represented as interface assertion \( S \)

logical formula with channel names as variables for streams
Continuous systems: the model

Sets of typed channels

\[ I = \{x_1 : T_1, x_2 : T_2, \ldots \} \]
\[ O = \{y_1 : T'_1, y_2 : T'_2, \ldots \} \]

syntactic interface

\( (I \triangleright O) \)

continuous data stream of continuous type \( M \)

\[ \text{ConSTREAM}[T] = \{IR_+ \to M\} \]

valuation of channel set \( C \)

\[ \text{CIH}[C] = \{C \to \text{ConSTREAM}[T]\} \]

interface behaviour for syn. interface \( (I \triangleright O) \)

\[ [I \triangleright O] = \{\text{CIH}[I] \to \varnothing (\text{CIH}[O])\} \]

interface specification

\[ p : I \cup O \to IB \]

represented as interface assertion \( S \)

logical formula with channel names as variables for continuous streams

Example: System interface specification

A transmission component TMC

\[
\begin{array}{c}
\text{in } x: \text{Data} \\
\text{out } y: \text{Data} \\
x \sim y
\end{array}
\]

\[
x \sim y \equiv (\forall m \in \text{Data}: \{m\} \bowtie x = \{m\} \bowtie y)
\]

Verification: Proving properties about specified components

From the interface assertions we can prove

- Safety properties

\[ \{m\}#y > 0 \land y \in TMC(x) \Rightarrow \{m\}#x > 0 \]

- Liveness/progress properties

\[ \{m\}#x > 0 \land y \in TMC(x) \Rightarrow \{m\}#y > 0 \]
Example: System interface specification - timing

A transmission component TMC

\[ TMC(\delta: \text{Time}) \]

\[ \begin{array}{l}
\text{in} & x: \text{Data} \\
\text{out} & y: \text{Data} \\
\end{array} \]

\[ \forall t \in \text{Time}, m \in \text{Data}: \]

\[ \{m\} \circ x \downarrow t \leq \]

\[ \{m\} \circ y \downarrow (t+\delta+1) \leq \]

\[ \{m\} \circ x \downarrow (t+\delta) \]
The principles of model based software and systems engineering

• The triangle
  ◊ Mathematical modeling concepts (denotational semantics)
  ◊ Logical system property calculus (deductive semantics)
  ◊ System implementation concept (operational semantics)

• Concept of unit and of compositions – for all elements of the triangle
  ◊ Modularity

• Concept of refinement

• Concept of semantic relationship and semantic coherence

• Concept of semantic dependencies
  ◊ Beyond tracing
Discrete systems: the modeling theory - probability

Sets of typed channels

\[ I = \{x_1 : T_1, x_2 : T_2, \ldots \} \]
\[ O = \{y_1 : T'_1, y_2 : T'_2, \ldots \} \]

syntactic interface

\((I \triangleright O)\)

data stream of type \(T\)

\[ \text{STREAM}[T] = \{IN\backslash\{0\} \rightarrow T^*\} \]

valuation of channel set \(C\)

\[ \text{IH}[C] = \{C \rightarrow \text{STREAM}[T]\} \]

interface behaviour for syn. interface \((I \triangleright O)\)

\[ [I \triangleright O] = \{\text{IH}[I] \rightarrow \text{PD}[\emptyset (\text{IH}[O])] \} \]

interface specification

\[ p: I \cup O \rightarrow IB \]

represented as interface assertion \(S\)

logical formula with channel names as variables for streams

Example: System interface specification - timing

A transmission component TMC

TMC(δ: Time)

\[
\begin{array}{ll}
\text{in} & \text{x: Data} \\
\text{out} & \text{y: Data}
\end{array}
\]

\[\forall t \in \text{Time}, m \in \text{Data}: \]
\[P(\{m\} \odot \! \! \! \! \downarrow \! \! \! \! x \downarrow t \leq \{m\} \odot \! \! \! \! \downarrow \! \! \! \! y \downarrow (t+\delta+1) \leq \{m\} \odot \! \! \! \! \downarrow \! \! \! \! x \downarrow (t+\delta)) > 0.99\]
A refined model of behavior: probability

• A model of functional correctness - a qualitative model

\[ F_{\text{logic}}: IH[I] \rightarrow \emptyset (IH[O]) \]

• A model of probability - quantitative model

\[ F_{\text{probabilistic}}: IH[I] \rightarrow \{ (\mu, Y): Y \subseteq IH[O] \} \]

where \((\mu, Y)\) denotes a probability distribution \(\mu\) over set \(Y\)
The overall system views

<table>
<thead>
<tr>
<th>Syntactic</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Logical</td>
</tr>
<tr>
<td>Black Box Interface</td>
<td>Syntactic interface</td>
</tr>
<tr>
<td>Glass Box</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td>Data flow graph</td>
</tr>
<tr>
<td>State</td>
<td>State space</td>
</tr>
</tbody>
</table>
## Functional requirements

<table>
<thead>
<tr>
<th>Syntactic</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black Box Interface</td>
<td></td>
</tr>
<tr>
<td>Syntactic interface</td>
<td>Logical interface behavior</td>
</tr>
<tr>
<td></td>
<td>Probabilistic interface behavior</td>
</tr>
<tr>
<td>Glass Box</td>
<td></td>
</tr>
<tr>
<td>Architecture</td>
<td></td>
</tr>
<tr>
<td>Data flow graph</td>
<td>Logical component interface behavior</td>
</tr>
<tr>
<td>Syntactic component</td>
<td>Probabilistic component interface behavior</td>
</tr>
<tr>
<td>interface</td>
<td></td>
</tr>
<tr>
<td>State</td>
<td></td>
</tr>
<tr>
<td>State space</td>
<td>Logical state machine</td>
</tr>
<tr>
<td></td>
<td>Probabilistic state machine</td>
</tr>
</tbody>
</table>
## The overall system views

<table>
<thead>
<tr>
<th></th>
<th>Syntactic</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Black Box Interface</strong></td>
<td>Syntactic interface</td>
<td>Logical interface behavior</td>
</tr>
<tr>
<td><strong>Glass Box</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Data flow graph</td>
<td>Logical component interface behavior</td>
</tr>
<tr>
<td></td>
<td>Syntactic component interface</td>
<td>Logical</td>
</tr>
<tr>
<td><strong>State</strong></td>
<td>State space</td>
<td>Logical state machine</td>
</tr>
</tbody>
</table>
System and its operational context

- Physical World
- Cyberspace Services & Data
- HMI
- System and its operational context
- Context System
System and its wider context

- Physical World
- HMI
- Cyberspace Services & Data
- Context system
- Larger context
- Business
- Development
- Cost
- Market
- Law
- Performance
- Execution
- Reuse
Main Quality Attributes

Quality in End-Use
- Usability
  - Operability
  - Learnability
- Reliability
- Security
- Safety
- Functional Suitability
  - Functional correctness
  - Functional appropriateness
  - Functional completeness
- Time behavior
  - Error protection
  - Accessibility
  - Response time
  - Throughput

Quality in Development and Evolution
- Maintainability
- Releasability
- Reusability
- Executability
- Supportability
- Quality in Business
  - Apprasability

Auxiliary Quality Attributes
- Performance
- Adaptability
- Portability
- Reviewability
Main Quality Attributes

Quality in End-Use
- Reliability
  - Operability
  - Learnability
- Functional Suitability
  - Functional correctness
  - Functional appropriateness
  - Functional completeness
  - Time behavior
- Security
  - Confidentiality
  - Integrity
- Safety
  - Economic damage risk
  - Health and safety risk
  - Environmental harm risk
- Maintainability
  - Analyzability
  - Modifiability
  - Verifiability
- Releasability
- Reusability
- Executability
- Supportability
- Apprasability

Quality in Development and Evolution
- Maintainability
- Releasability
- Reusability
- Executability
- Supportability
- Apprasability

Quality in Operation
- CPU Consumption
- Memory Consumption
- Co-existence

Quality in Business
- Error protection
- Accessibility
- Response time
- Throughput
- Analyzability
- Testability
- Reviewability
- Modifiability
- Installability
- Reusability
- Configurability
- Co-existence

Auxiliary Quality Attributes
- Adaptable
- Portability
- Performance
Classification

• Functional requirements: logical and probabilistic interface behavior (including faults):
  ◊ functional features
  ◊ safety
  ◊ reliability
  ◊ ...

• Architectural requirements: logical and probabilistic sub-system interface behavior (including faults)
  Quality requirements such as:
  ◊ Performance
  ◊ Security

• Requirements related to system context
  ◊ Usability
  ◊ Business - Return on investment
Conclusion

Probability
• Probabilistic system models and specifications are refinements of nondeterministic system models and specifications
• A rich set of so-called “non-functional” properties is captured by probabilistic interface specifications and thus become functional

Time
• Time-aware system models and specifications are refinements of non-time-aware system models and specifications
• For time critical systems so-called “non-functional” timing properties can be captured by time-aware interface specifications and thus become functional

Performance
• There are two concepts of performance:
  ◊ response time (in the case of non-time critical functionality)
  ◊ efficient utilization of resources
• Response time is a functional property captured by time-aware probabilistic interface specifications
Interesting questions

• **Systems show probabilistic behavior!**
• **Is software probabilistic?**
  ◊ If no random concepts are part of the program?
  ◊ If the operational context is probabilistic?
  ◊ If the execution platform is understood to be “probabilistic”?
• **Remark on refinement of probabilistic specifications, architecture models and implementations**
  ◊ A system behavior specification with a specified probabilistic distribution cannot be further refined!
  ◊ General probabilistic specifications define sets of propabilistic behaviors with specific probability distributions.