Requirements Engineering I

Chapter 6

Specifying with Natural Language



© 2025 Martin Glinz

Chapter roadmap



Problems and rules 6.1

Good to know

ML/AI tools and assistance 6.5

Helpers for writing and analyzing

Phrase templates 6.2

The key means for writing better NL requirements

User stories 6.3

The main work product in agile

Form templates 6.4

Help with structured work products

© 2025 Martin Glinz

Natural language for expressing requirements

The system shall ...

The oldest...

...and most widely used way

- taught at school
- extremely expressive

But not necessarily the best

- Ambiguous
- Imprecise
- Error-prone
- Careful writing and reading needed



Michelangelo's Moses (San Pietro in Vincoli, Rome)
Moses holds the Ten Commandments in his hand:
written in natural language

© 2025 Martin Glinz

6.1 NL requirements: problems and rules

Read the subsequent requirements. Any findings?

"For every turnstile, the total number of turns shall be read and archived once per day."

"The system shall produce lift usage statistics."

"Never shall an unauthorized skier pass a turnstile."

"By using RFID technology, ticket validation shall become faster."

"In the sales transaction, the system shall record the buyer's data and timestamp the sold access card."

© 2023 Martin Glinz

Some rules for specifying in natural language

[Rupp et al. 2020]
[Goetz&Rupp 2003]

- Use active voice and defined subjects
- Build phrases with complete verbal structure
- Use terms as defined in the glossary
- Define precise meanings for auxiliary verbs (shall, should, must, may,...) as well as for process verbs (for example, “produce”, “generate”, “create”)
- Check for nouns with unspecific semantics (“the data”, “the customer”, “the display”,...) and replace where appropriate
- When using adjectives in comparative form, specify a reference point: “better” → “better than”

© 2024 Martin Glinz

More rules

- Scrutinize all-quantifications: “every”, “always”, “never”, etc. seldom hold without any exceptions
- Scrutinize nominalizations (“authentication”, “termination”...): they may conceal incomplete process specifications
- State every requirement in a main clause. Use subordinate clauses only for making the requirement more precise
- Attach a unique identifier to every requirement
- Structure natural language requirements by ordering them in sections and sub-sections
- Avoid redundancy where possible

© 2020 Martin Glinz

All-quantification and exclusion

- Specifications in natural language frequently use all-quantifying or excluding statements without much reflection:

  "When operating the coffee vending machine, the user shall always be able to terminate the running transaction by pressing the cancel key."

  Also when the coffee is already being brewed or dispensed?

- ⇨ Scrutinize all-quantifications ("every", "all", "always"...) and exclusions ("never", "nobody", "either – or",...) for potential exceptions
- ⇨ Specify found exceptions as requirements

© 2025 Martin Glinz

Dealing with redundancy

- Natural language is frequently (and deliberately) redundant

  → Secures communication success in case of some information loss

- In requirements specifications, redundancy is a problem
  - Requirements are specified more than once
  - In case of modifications, all redundant information must be changed consistently
- Make redundant statements only when needed for abstraction purposes
- Avoid local redundancy: “never ever” → “never”

© 2025 Martin Glinz

6.2 Phrase templates

[Mavin et al. 2009]
[Rupp et al. 2020]
[ISO/IEC/IEEE 2018]

Use templates for creating well-formed natural language requirements

Typical template:

[<Condition>] <Subject> <Action> <Objects> [<Restriction>]

Example:

When a valid card is sensed, the system shall send the command ‘unlock_for_a_single_turn’ to the turnstile within 100 ms.

© 2024 Martin Glinz

EARS

[Mavin et al. 2009]

EARS (Easy Approach to Requirements Syntax) provides a set of templates:

- Ubiquitous requirements (no restrictions)

  "The control system shall prevent engine overspeed."

- Event-driven requirements (WHEN)

  "When continuous ignition is commanded by the aircraft, the control system shall switch on continuous ignition."

- Unwanted behavior (IF-THEN)

  "If the computed airspeed fault flag is set, then the control system shall use modeled airspeed."

© 2021 Martin Glinz

EARS – 2

- State-driven requirements (WHILE)

  "While the aircraft is in-flight, the control system shall maintain engine fuel flow above XX kg/s."

- Optional features (WHERE)

  "Where the control system includes an overspeed protection function, the control system shall test the availability of the overspeed protection function prior to aircraft dispatch."

- Complex requirements are expressed by combining keywords:

  "While the aircraft is on-ground, when reverse thrust is commanded, the control system shall enable deployment of the thrust reverser".

© 2021 Martin Glinz

6.3 User stories

[Cohn 2004]

- A single sentence about a requirement
- Written from a stakeholder's perspective
- Accompanied by acceptance criteria
- Acceptance criteria make the story more precise

Standard template:

As a <role> I want to <my requirement> so that <benefit>

© 2025 Martin Glinz

A sample user story

As a skier, I want to pass the chairlift gate so that I get access without presenting, scanning or inserting a ticket at the gate.

Author: Dan Downhill Date: 2013-09-20 ID: S-18

© 2017 Martin Glinz

Sample acceptance criteria

Acceptance criteria:

- Recognizes cards worn anywhere in a pocket on the left side of the body in the range of 50 cm to 150 cm above ground
- If card is valid: unlocks turnstile and flashes a green light for five seconds or until the turnstile is moved
- If card is invalid: doesn't unlock gate and flashes a red light for five seconds
- Time from card entering the sensor range until unlock and flash red or green is less than 1.5 s (avg) & 3 s (max)
- The same card is not accepted twice within an interval of 200 s

© 2017 Martin Glinz

Mini-Exercise: Writing a user story

Consider the chairlift access control case study.

Write a story from a skier's perspective about buying a day card.

© 2024 Martin Glinz

6.4 Form templates

- Provide a form with predefined fields to be filled in.
- Examples:
  - Use case templates (see Chapter 7)
  - Quality requirements template

A form template for specifying measurable quality requirements [Glinz et al. 2024]

Template		Example
ID	<Number of requirement>	R137.2
Goal	<Qualitatively stated goal>	Confirm room reservations immediately
Scale	<Scale for measuring the requirement>	Elapsed time in seconds (ratio scale)
Meter	<Procedure for measuring the requirement>	Timestamping the moments when the user hits the "Reserve" button and when the app has displayed the confirmation. Measuring the time difference.
Minimum	<Minimum acceptable quality to be achieved>	Less than 5 s in at least 95% of all cases
OK range	<Value range that is OK and is aimed at>	Between 0.5 and 3 s in more than 98% of all cases
Desired	<Quality achieved in the best possible case>	Less than 0.5 s in 100% of all cases

© 2024 Martin Glinz

6.5 ML/AI tools and assistance

Until recently, both writing and analyzing natural language requirements were genuinely human tasks.

- The rise of machine learning (ML) based tools has enabled the development of automated analysis tools
- The advent of GenAI has enabled automated writing of well-structured NL requirements from unstructured collections of information.
- Critically reviewing the results still is a human task

© 2025 Martin Glinz