Quality makes products which do not return and customers who do

Erik van Veenendaal
www.erikvanveenendaal.nl
Recognize this?

- **Tester:**
  “Can’t test this, not clear, not unambiguous”

- **Requirements engineer:**
  “What is a good testable requirement?”

- **Tester:**
  “Uuuuhhhh…. SMART” (or INVEST)

- **Requirements engineer:**
  “Let’s define ‘what are the requirements for requirements?’”

However….

- We are one of the main stakeholders, planning, risk analysis, (test) designs and code are based upon **requirements**
- We are involved in **requirements** reviews, what level of quality is reasonable?
- (Test) designs may even be used as **requirements**
- Sometimes (in Agile) we identify and specify **requirements** and acceptance criteria
  - We have a major interest in **requirements** and are heavily involved!
T-Shaped Testers

---

**Business Analysis**

**Domain**

**Development**

**Testing**

---

Requirements Engineering for Testers

- 08:15 – 08:45 Introduction to requirements engineering
- 08:45 – 09:15 Requirements process
- 09:15 – 09:45 What are good requirements? (1)
- 10:00 – 10:45 What are good requirements? (2)
- 10:45 – 11:15 Exercise “writing requirements”
- 11:15 – 11:30 Reviewing requirements
- Evaluation and Closure
Learning Objectives

- Understand the importance of requirements
- Have an overview of requirements engineering process
- Learn a structured approach for writing good requirements in a natural language
- Provide practical ideas for writing better requirements
- Be able to organize and participate in requirements reviews
- **Note:** in Agile requirements come as user stories

The Challenge

- To capture the need “completely” and “unambiguously” without resorting to specialist jargon, thus understandable by our stakeholders
- Requirements are the basis for:
  - Project (release) planning
  - Trade-off (priority setting)
  - Development
  - Testing
Project Success Factors….


Basic Communication Model

Sender

Message

Receiver

Encodes

Joint code

Decodes
What is a Requirement?

A requirement is a capability or condition to which the system must conform

... a capability needed by the user to solve a problem or achieve an objective

... a capability that must be met or possessed by a system to satisfy a contract, specification, standard or other formally imposed documentation

... a statement of intent that describes something the system needs to do for some user

Three Types of Requirements

- **Functional requirements** are things the product must do
  - The product shall produce an updated schedule
  - As a <student>, I want <to be able to buy a parking pass> so I can <get to school quickly>

- **Non-functional requirements** (e.g., ISO25001) are the properties that the product must have
  - The product shall determine ... in less than 0.25 seconds
  - As a <member of the public> I want <the website to adequately cope with high loads> so I can <purchase a ticket quickly for a highly subscribed event>

- **A constraint** is a restriction on the scope or design of the product
  - The product shall run on the ... platform
A main principle……

How much documentation is enough?

- Requirements process is context dependent

  - User requirements – problem domain
    - State what the stakeholders want to achieve through use of the system. Avoid reference to any particular solution. “The user shall be able to…..”

  - System requirements – solution domain
    - State abstractly how the system will meet the stakeholder requirements. Avoid reference to any particular design. “The product shall …..”

- Agile / V-model / Outsourcing
- Business / Project / Product / Human factors

More Definitions….  

- Requirements Engineering
  - A systematic approach to gathering, organizing, and documenting the requirements of the system

- Requirements Management
  - A process that establishes and maintains agreement between the customer and the project on the changing requirements of the system
  - Agile: Managing the Backlog by Product Owner
Requirements Process (1)

1. Kick-off phase
   - Objective, scope, stakeholders, business case
   - Check: Are things clear enough to start?

2. Requirements gathering (quantity-based)
   - Functional, Non-functional, Constraints
   - Various gathering / elicitation techniques
   - Building the backlog

3. Requirements specification (quality-based)
   - Templates, rule set, level of detail needed
   - Product backlog grooming / refinement

From Quantity to Quality
Backlog Grooming (Agile)

- Bring user stories to an acceptable level of detail & maintaining a rolling backlog projection
- Plan 10% of each sprint to be spent grooming the backlog
- Top ordered backlog items are well understood and easily selected in sprint planning
- Actions in meeting
  - Provide feedback on clarity, quality, acceptance criteria, dependencies, ranking of stories
  - Identify holes for which user stories must be written
  - Break epics into stories

Requirements Engineering for Testers

- 08:15 – 08:45 Introduction to Requirements Engineering
- 08:45 – 09:15 Requirements Process
- 09:15 – 09:45 What are good requirements? (1)
  - 10:00 – 10:45 What are good requirements? (2)
  - 10:45 – 11:15 Exercise “writing requirements”
  - 11:15 – 11:30 Reviewing requirements
  - Evaluation and Closure
What is needed ….

- If we could look into each other’s brains, we wouldn’t need documentation …
- Documentation helps us to communicate

- Be careful, words will not be enough!
  - Formal / informal language
  - Different interpretations

"I didn't say he killed his wife“

"I didn't say he killed his wife"
"I didn't say he killed his wife"
"I didn't say he killed his wife"
"I didn't say he killed his wife"
"I didn't say he killed his wife"
"I didn't say he killed his wife"
What are “good” requirements?

Identify at least five “rules” that determine whether a requirement is a good (or “poor”) requirement

Consider !!
- Individual requirements (the “sentence”)
- Requirements attributes

Excercise Radio Watch (1)

- Study the requirement specification for the new Radio Watch
- Make comments (find defects) based on what you have learned so far e.g., attributes, rules, ….
The development of the innovative watch and music system shall raise the profit of the company by 10% within 6 months after delivery to the whole sale dealers.

The watch shall have no tic sound. Because people don't like to be waked up, the new radio watch shall wake up sleeping people more smoothly than any other watch.

The radio watch can operate with FM bandwidth only.

The radio watch will provide the user with data and time information.

The radio watch will provide the hours and minutes in digital format.

The watch shall be small and lightweight, because the users want to use it also on travel. Therefore the watch shall be usable with different power voltages. To achieve a minimum weight the watch must not contain any iron parts.

The watch shall be able to wake its environment with music or a wake-up call.

The watch shall be very easy to use. The snooze key shall be easy accessible.

The watch shall be highly reliable. It shall resist a heavy knock of a sleeper and a fall from 30 cm.

The radio watch shall be very efficient in energy consumption. The watch system may only be operated with batteries of type S434.

The radio watch must be able to save settings for wake up time.
Requirements Engineering for Testers

- 08:15 – 08:45 Introduction to Requirements Engineering
- 08:45 – 09:15 Requirements Process
- 09:15 – 09:45 What are good requirements? (1)

- 10:00 – 10:45 What are good requirements? (2)
- 10:45 – 11:15 Exercise “writing requirements”
- 11:15 – 11:30 Reviewing requirements
- Evaluation and Closure

The purpose of the product

- The user problem *(no more than 1 page)*
  - A short description of the situation that triggered the development effort
  - Describe the work that should be improved
- Goals of the project
  - What will the product do? (purpose)
  - What is the business advantage?
  - How will you measure the advantage?
  - Statement of needs on a high-level

Get stakeholders commitment on this !!
Purpose Example (A’dam Metro)

**Purpose:** To sell metro tickets more efficiently (faster) than currently.

**Rational:** To increase sales and reduce cueing while buying metro tickets.

**Acceptance Criteria:** The product will hand out tickets 30% faster than the current system. This improvement shall be achieved on all priority 1 stations at peak hours.

---

Use Cases / EPICs

- Start with the context diagram
- Use cases give users a convenient way to partition the product
- One or more use cases per business event
  - also consider ‘misuse cases’, e.g., for security req.
- Six step scenario’s are a great starting point
  - Name
  - Actor (user)
  - Short description (‘happy day scenario’)
  - Pre conditions
  - Post conditions
Hierarchy and Traceability

Use Case Example (A’dam Metro)

**Use Case:** Traveller buying a ticket.

**Actor:** Traveller

1. The traveller offers destination, type of ticket and payment to the product
2. The product checks whether the payment is ok for the chosen destination and type of ticket
3. The product checks whether the network is operational for the chosen destination.
4. The product submits ticket and if necessary change.
5. The product stores the transaction
Requirements Example (A’dam Metro)

**Use Case step 2.** The product checks whether the payment is ok for the chosen destination and type of ticket

**Requirements:**

2.1 The product shall establish that the payment consists of legally valid money

2.2 The product shall calculate the lowest fare for the destination considering day of week and time

2.3 The product shall compare the travellers’ payment with the calculated payment

2.4 The product shall provide feedback in case the payment is not sufficient.

or from EPICs to User Stories

“As a customer, I want to have wishlists so I can come back to buy products”

“As a customer, I want to save a product to a wishlist”

“As a customer, I want to view my wishlists so I can buy items from it”
Example

Requirements cards

<table>
<thead>
<tr>
<th>Requirement #</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement Type</td>
<td>Use case</td>
</tr>
<tr>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>Rationale</td>
<td></td>
</tr>
<tr>
<td>Source</td>
<td></td>
</tr>
<tr>
<td>Acceptance Criteria</td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td></td>
</tr>
<tr>
<td>Supporting material: annotation, conversation, examples</td>
<td></td>
</tr>
</tbody>
</table>
### Requirements attributes (1)

- **ID**
  - To allow traceability

- **Requirements Type**
  - Allows req.’s to be sorted, grouping allows the requirements to be checked on completeness and for conflicts, e.g., by non-functional, by business process

- **Use case**
  - For traceability and change control purposes
  - Again for grouping etc.

- **Description**
  - The intent of the requirement (may initially be ambiguous)
  - the stakeholders’ whishes & needs

### Requirements attributes (2)

- **Rationale**
  - Reason behind the requirement’s existence. Helps to clarify and understand the requirement and to identify ‘gold plating’ req.’s.

- **Priority**
  - Measure of the business value and importance. For negotiation, but also for risk-based testing

- **Acceptance criteria**
  - To make the requirement measurable and testable

- **Source**
  - Name of the person who raised the requirement, or document.

- **Size**
  - Number of story points
Acceptance Criteria

- We have to be able to tell whether a solution completely satisfies, or fits, a requirement
- To make requirements measurable / testable
- To define boundaries
- To reach consensus (with the stakeholder)
- To allow accurate planning & estimation
- In practice very important for non-functional requirements
  
  - 90% of the customers must be able to get the correct ticket from the product in no more than 25 seconds

Example Acceptance Criteria

Requirement / User Story
- As a student, I want to be able to buy a parking pass so I can get to school quickly

Acceptance Criteria (5 to 8)
- The student will not receive the parking pass if the payment is insufficient
- One can only buy a parking pass to the school parking lot if the person is a registered student
- The student can only buy one parking pass each month
- .....
User Story & Acceptance Criteria

As a tutorial attendee, I want to be able to register with techwell online, so I can register quickly and cut down on paperwork.

Questions by the tester might include

- What information needs to be collected to allow a user to register?
- Where does this information need to be collected/delivered?
- Can the user pay online as part of the registration process?
- Does the user need to be sent an acknowledgment?

Acceptance Criteria

- A user cannot submit a form without completing all the mandatory fields.
- Information from the form is stored in the registrations database.
- Protection against spam is working.
- Payment can be made via credit card.
- An acknowledgment email is sent to the user after submitting the form.

Writing Guidelines

To write simple is as difficult as to be good

- Short and concise sentences and paragraphs
- One requirement per sentence
  - no compound requirements, a single verb
- Consistent terminology (homonyms)
- Avoid generalizations
- Use ‘must’, ‘can’ and similar words carefully
  - ‘shall’ is better, ‘can’ indicates options
- No solutions or design
- Directive language
- No pronouns
Use Templates

- **The** `<stakeholder>` *shall be able to* `<capability>`
  - The order clerk shall be able to raise an invoice

- **As a** `<role>`, *I want* `<activity>` *so that* `<business value>`
  - As a job seeker, I want to search for a job, so I can advance my career

- **The** `<product>` *shall be able to* `<action>` `<entity>`
  - The launcher shall be able to launch missiles

- **The** `<product>` *shall* `<function>` `<object>` *every* `<performance>` `<unit>`
  - The coffee machine shall produce a hot drink every 10 seconds

Requirements Rule Set

- **Usefull** set of agreements
- Specify the contents and format of a requirement (and requirements document)
- More objective, less discussion
- Applied during specification and reviewing
- Organization specific
- Rules for tracing, format and content
Examples of Rules (1)

- Identification
- Valuable / purpose
- Changes
- Grouping
- Uniqueness
- Consistency
- Annotation
- Language
- No design
- Knowledge responsible (source)

All forms of annotation, comments, notes, suggestions, examples, or other items not part of the formal requirement shall be clearly indicated as such. This will be documented by using the attribute ‘additional information’.

Examples of Rules (2)

- Detail
- Brief / Small
- Unambiguous
- Priority
- Rationale
- Compound
- Independent
- Technically achievable
- Testable

Req.’s shall be unambiguous to the intended readership. Req.’s shall have only one interpretation. For example the word shall is used and not the word should. Words like can shall only be used when more than one option is available. Directive language (active voice) shall be used, e.g., specifies and not can specify.
Interpretation: Unambiguous

- **To be part of the backlog**
  - The requirements shall be at the level of unambiguousness to allow product team level decisions to be taken.

- **To be part of the sprint planning**
  - The requirements shall be at the level of unambiguousness to allow for estimation in terms of effort and time.

- **To be part of the sprint**
  - The requirements shall provide enough information to allow for the execution of individual deliverables and tasks (e.g., detailed design, test design).

Excercise Radio Watch (2)

1. Create your requirements **standard**
   - select the **rules** you will use
   - **attributes** that you will use
   - select a description **template**

2. Rewrite some of the requirements for the new **Radio Watch**
   - Make concrete improvement suggestions
   - Watch out for fuzzy terms
   - Use the requirements **card**
Requirements Engineering for Testers

- 08:15 – 08:45 Introduction to Requirements Engineering
- 08:45 – 09:15 Requirements Process
- 09:15 – 09:45 What are good requirements? (1)
- 10:00 – 10:45 What are good requirements? (2)
- 10:45 – 11:15 Exercise “writing requirements”
- 11:15 – 11:30 Reviewing requirements
- Evaluation and Closure

Why Verification and Validation?

- Efficient and effective way to find defects
  - ambiguous, consistency, completeness, compound, ...
- Many defects have already been made before design has started
  - 50% “requirements related”
- Early defects are highly important
  - defects have the characteristic to multiply themselves top-down
  - cost of rework rises (exponentially)
Core Agile Practice: Reviews

Communication & Feedback
Customer collaboration

- Walkthrough / Pair Inspection / Informal Reviews
- Priority to the profitable
- Apply review practices that make the difference

Requirements reviews - types

- **Walkthrough** (with stakeholders)  
  - product owner will guide the group through the requirements
  - common understanding, knowledge sharing, consensus, information gathering

- **Inspection** (with fellow engineers)  
  - check against rules
  - individual process to find defects
  - using checklists

Validation
Verification
Walkthrough

- **Planning** (scrum master): no formal entry criteria
- **Preparation** (reading): preparing questions
- **Kick-off** at the beginning of the meeting: objective
- **Meeting**: author provides explanation (e.g., scenario’s, prototypes, use cases)
  - Scribe to records findings
  - Scrum master manages the process (chair)
- **Rework/exit**: not formal, author continues the work
(Pair) Inspection

- Planning/Kick-off: requirements, rules, objective
- Preparation: individual, checking not reading
- Meeting: defects explained, discussion, logging? improvement suggestions?
- Rework: by author, log as a 'checklist'
- Follow-up/Exit: check updated requirements
  - 1 in 10 defects is not addressed correctly
    (Source: Les Hatton)

Lessons Learned

- What will YOU take home?
- Action list
- Make this explicit
- Awareness!!
Things consider tomorrow

From previous groups

1. Start with a purpose statement & context diagram
2. Structure (feature, epic, story)
3. Introduce requirements attributes (cards)
4. Use acceptance criteria to fine-tune req.’s
5. Use templates
6. Define and use requirements rules
7. First quantity, than quality
8. Introduce practical reviews

www.erikvanveenendaal.nl
erik@erikvanveenendaal.nl
@ErikvVeenendaal
Any questions.....?

Thank you !!