Requirements Gathering Rules
Basic guidelines for any project

Alain Leroy
August 2, 2012

BDPA National Technology Conference
Baltimore, MD
Manager at Deloitte Consulting with over 14 years of industry and consulting experience in driving business transformation at Fortune 500 Technology and Health Care companies.

• Previous experience:
  – Process Improvement Manager and IT Account Manager at WellPoint
  – Senior Strategist and Branding Manager at IBM Software Group
  – IT Consultant at PA Consulting Group and IBM Global Services
  – Software developer at start-up company

• Education:
  – BS in Computer Science (1997) from Rochester Institute of Technology

• Certifications:
  – Six Sigma Black Belt
  – Project Management Professional (PMP®)
  – Scrum Master (CSM)
Introductions

1. What’s your name?
2. Where do you live / work?
3. What’s your current role?
4. Do you have any previous requirements experience?
5. What do you hope to get out of this session?
Agenda

• Art of Writing Good Requirements
• Requirements Gathering Checklist
• Comparison of Methods
• Key Takeaways & Tips
• Q & A
Learning objectives

After completing this session you will:

• Understand business requirements development
• Write high-level and detailed requirements
• Overview requirements tracking tools
• Know basic rules for effective requirements analysis
Agenda

- **Art of Writing Good Requirements**
  - Requirements Gathering Checklist
  - Comparison of Methods
  - Key Takeaways & Tips
  - Q & A
At the start of a project, it is very important to quickly and efficiently gather expectations, filter through them and develop a manageable list of requirements.
Is this common at your company?
There is no silver bullet
High-level requirements gathering process

Core activities for any project

- Identify stakeholders
- Determine elicitation approach
- Plan requirements gathering activities

- Write requirements

- Review requirements
- Track requirements
- Accept requirements
- Change requirements
## Where requirements gathering fits in the SDLC

<table>
<thead>
<tr>
<th>Phase</th>
<th>Sub phases</th>
<th>Objectives</th>
</tr>
</thead>
</table>
| Inception   | Initiate   | • Kick off the project  
• Gain agreement with stakeholders on project scope  
• Create an initial plan for achieving project scope |
|             | Plan       | • Plan activities for project execution  
• Resulting project plans provide a detailed-level view of the project |
| Elaboration | Requirements | • **Perform requirements solicitation**  
• **Document functional and non-functional requirements**  
• **Produce requirements specification for use on the project** |
|             | Design     | • Design product or solution to meet client's specifications  
• Initiate development of supporting change management artifacts or training materials |
|             | Development | • Complete development of product or solution components  
• Self validate components meet client expectations  
• Develop test cases to support various types of quality control testing  
• Develop transition support materials |
|             | Test       | • Plan for and execute various types of testing  
• Complete testing of product or solution |
| Transition  | UAT        | • Plan and complete user acceptance testing of the solution |
|             | Train      | • Plan for and deliver end-user training for the product or solution |
|             | Deploy     | • Deploy solution in production  
• Assess effectiveness of change management approach |
|             | Close      | • Validate all project activities completed  
• Validate solution formally accepted by client  
• Validate all project documentation archived |
Requirements elicitation activities

• Identify stakeholders
• Determine approach
• Plan activities

Start here....

Elicitation → Development → Management
Stakeholder identification

Before requirement development begins, identify all the stakeholders for the project

- Understand how stakeholders will be affected by the project’s outcome
- Understand what stakeholder’s involvement or influence will be in the project

Without this planning you may miss key stakeholders, requiring additional sessions to get their inputs.
Elicitation approach

Understanding the purpose and success criteria will help decide what technique should be used

**Elicitation techniques:**

- Observation
- Interviews with Stakeholders
- Existing documentation
- JAD sessions
- User stories
- Use cases

There is no one perfect method for eliciting requirements.
Requirements planning

Identify and plan requirements gathering activities to effectively manage project requirements from Elaboration to Deployment

• Onboarding and training resources to perform requirement elicitation and development activities
• Detailing the work plan
• Tailoring the processes and templates to support the project team
• Setting up any tools required to support the capture of requirements
Requirements development activities

- Elicitation
- Development
- Management

- Write requirements
The author’s role & responsibilities

Writing requirements takes great skill and, like writing code, the end result is usually cleaner and more consistent if there’s a single author.

• Translate the customers wants and needs into requirements for the developer
• Translate technical jargon into business language
• Performing business modeling
• Capturing high level and detailed requirements
• Completing requirements documentation
The author’s dilemma....

The imprecision of communication plays major role when interacting with all stakeholders
Writing good requirements

A good requirement is a simple, complete, well-structured sentence

• Statement of need, something a user or stakeholder wants
• Specification of what should be implemented
• Description of how the system should behave
True or false?

A requirement is not an assumption?
## Characteristics of good requirements

<table>
<thead>
<tr>
<th>Complete</th>
<th>Requirements should be as complete as possible—no open-ended requirements.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testable</td>
<td>Must be able to create a test for all requirements.</td>
</tr>
<tr>
<td>Consistent</td>
<td>Requirements must be consistent with each other—no conflicts.</td>
</tr>
<tr>
<td>Design Free</td>
<td>Software requirements should be specified in the business perspective rather than the software perspective.</td>
</tr>
<tr>
<td>Unambiguous</td>
<td>Use &quot;shall&quot; and other related words. Don’t be wishy-washy.</td>
</tr>
</tbody>
</table>
**Will, Must, Shall, Should….**

<table>
<thead>
<tr>
<th>Verb</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shall</td>
<td>Indicative of a requirement</td>
<td>A report shall be needed to list everyone in the group with blue eyes.</td>
</tr>
<tr>
<td>Must</td>
<td>Critical Requirement Constraint</td>
<td>There must be a new eligibility interface designed for the new drug vendor.</td>
</tr>
<tr>
<td>Will**</td>
<td>Declaration of fact</td>
<td>The group termination reports SR9999 will be changed to reflect the new term date format.</td>
</tr>
<tr>
<td>Should</td>
<td>Goal</td>
<td>Use of patch 334555 should improve performance of batch billing 50%.</td>
</tr>
</tbody>
</table>

**Will statements can also apply an assumption, so requirements must be worded clearly to avoid confusion.**
Exercise

Write a single, simple requirement for any of the following situations.

1. Your evening meal
2. A vacation home
3. A used car
Telephone operators should process 100 calls per hour

The team of telephone operators should be able to complete up to 100 reservations per hour during peak hours

Quite often what seems clear at first glance isn’t necessarily so.
## Business requirement types

There are many different types of business requirements

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functional</td>
<td>New features of the system or how existing features will be modified</td>
</tr>
<tr>
<td>Usability</td>
<td>Effectiveness, efficiency, satisfaction, user-friendly</td>
</tr>
<tr>
<td>Reliability</td>
<td>Availability, accuracy, defect rates</td>
</tr>
<tr>
<td>Performance</td>
<td>Response time, throughput, capacity, scalability, resource utilization</td>
</tr>
<tr>
<td>Supportability</td>
<td>On-going supportability</td>
</tr>
<tr>
<td>Security</td>
<td>Security, integrity, privacy</td>
</tr>
<tr>
<td>Data</td>
<td>All data elements used with the product</td>
</tr>
<tr>
<td>Training</td>
<td>End user training or documentation (user manuals, help guides)</td>
</tr>
<tr>
<td>Business Rules</td>
<td>Legal disclaimers, warranties, copyright or patent notice, watermark, trademark, branding, logo compliance, corporate policies, practices, mandates, performance guarantees</td>
</tr>
<tr>
<td>User</td>
<td>Task that the user must perform with the product</td>
</tr>
<tr>
<td>Out of Scope</td>
<td>Requirements identified as out of scope</td>
</tr>
</tbody>
</table>
Requirements management activities

- Elicitation
- Development
- Management

- Review
- Track
- Accept
- Change
Reviews

Reviews help prevent defects in future phases or implementation

Why perform reviews?
✓ Feedback
✓ Prevention of defects
✓ Improvement opportunities

Types of reviews:

• **Informal reviews**
  – Occurs anytime
  – Reviewed by any team member

• **Formal reviews**
  – Twice in SDLC-Waterfall
    – After requirements phase
    – After analysis phase
  – Reviewed by team member
  – Frequently in SDLC-Agile
    – At beginning & end of iteration
    – During release planning
    – Reviewed by product owner
Traceability

The primary purpose of traceability is to track an element or requirement from its creation through analysis and design, development, testing, and finally through deployment.

- Requirements Traceability Matrix (RTM) is a common tool used for requirements bi-directional traceability across the development lifecycle.
- In order to create the RTM, it is important to establish the requirements hierarchy, per the project’s Requirements Plan, which the project team uses to manage traceability links.
Elements of the Requirements Traceability Matrix (RTM)

<table>
<thead>
<tr>
<th>Requirement ID</th>
<th>Requirement ID refers to the Requirement number or an ID as captured in the Software Requirements Specification (SRS) document</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Process Model</td>
<td>The Requirements that would map to the Business Process Models as part of the As-is and/or to-be BP models</td>
</tr>
<tr>
<td>Use Case Name/ID</td>
<td>Name and ID of the Use Case that addresses the requirement</td>
</tr>
<tr>
<td>Design Document Name</td>
<td>This column should include the name of the relevant design artifacts that would map to these requirements</td>
</tr>
<tr>
<td>Component Name</td>
<td>The Component Name refers to the name of the solution software code developed as part of the solution development tasks which maps to the specific Design artifact and the Requirements ID</td>
</tr>
<tr>
<td>Unit Test Case ID</td>
<td>The Test Case ID provides information pertaining to the high level Test discipline documents. This column show contain the test case that was executed for the requirement during the Unit Test phase</td>
</tr>
<tr>
<td>System Test Case ID</td>
<td>The Test Case ID provides information pertaining to the high level Test discipline documents. This column show contain the test case that was executed for the requirement during the System Test phase</td>
</tr>
</tbody>
</table>
Acceptance - Definition of done

All stakeholders must have a common understanding of “done”

What does it mean to be “done?”

Basic definition of “done” means the requirement is designed, built, tested and accepted by stakeholder, but other things may need to be completed before the requirement is “done”
Agenda

• The Art of Writing Good Requirements

• Requirements Gathering Checklist
  • Comparison of Methods
  • Key Takeaways & Tips
  • Q & A
Requirements gathering checklist

1. **Elicitation**
   - Have you identified all the stakeholders?
   - What methods will you use to elicit requirements?
   - Do you understand the stakeholders needs, wants or constraints?
Requirements gathering checklist

1. Elicitation
   • Have you identified all the stakeholders?
   • What methods will you use to elicit requirements?
   • Do you understand the stakeholders needs, wants or constraints?

2. Development
   • Have you documented the high level requirements based on the stakeholders needs and project scope?
   • Have you documented the detailed requirements
   • Are there system or user requirements or both?
   • Have you logically grouped the requirements?
   • Have you verified the requirements are defect free?
   • Have you formulated the requirements in simple, complete, well structured sentences?
   • Have you checked for missing requirements?
Requirements gathering checklist

1. Elicitation
   • Have you identified all the stakeholders?
   • What methods will you use to elicit requirements?
   • Do you understand the stakeholders needs, wants or constraints?

2. Development
   • Have you documented the high level requirements based on the stakeholders needs and project scope?
   • Have you documented the detailed requirements
   • Are there system or user requirements or both?
   • Have you logically grouped the requirements?
   • Have you verified the requirements are defect free?
   • Have you formulated the requirements in simple, complete, well structured sentences?
   • Have you checked for missing requirements?

3. Management
   • Have you validated the requirements against the project charter, scope, stakeholders needs and wants and project success criteria?
   • Have the requirements been reviewed for acceptance
   • Where the requirements traced from inception to implementation?
   • Did you obtain all stakeholder approvals?
   • Are material changes being managed?
Agenda

• Art of Writing Good Requirements
• Requirements Gathering Checklist
• **Comparison of Methods**
  • Key Takeaways & Tips
  • Q & A
Comparison of methods

Requirements gathering tasks differ significantly from a traditional Waterfall to Agile. Agile is focused on flexibility and speed, while Waterfall prefers planning and predictability.

<table>
<thead>
<tr>
<th>Overview</th>
<th>Agile</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is iterative</td>
<td>• Prioritizes individuals &amp; interactions</td>
<td>• Is linear &amp; sequential</td>
</tr>
<tr>
<td></td>
<td>• Favors adaptability</td>
<td>• Prioritizes processes &amp; tools</td>
</tr>
<tr>
<td></td>
<td>• Produces working software in increments over 2-week “sprints”</td>
<td>• Favors predictability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Spans months from planning to final product</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Agile</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines high-level requirements or “Epics”</td>
<td>• Defines and refine “User Stories” at sprint planning for each Sprint</td>
<td>• Defines comprehensive list of requirements at outset of development lifecycle</td>
</tr>
<tr>
<td>in product &amp; release plans</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defines “User Stories” at sprint planning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for each Sprint</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Documentation</th>
<th>Agile</th>
<th>Waterfall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefers working software</td>
<td>• Creates and iterate living, breathing user stories</td>
<td>• Prefers comprehensive documentation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Creates static requirements documentation at beginning of project</td>
</tr>
</tbody>
</table>

Agile terms

Multiple levels of functional detail within Agile

- **User Stories.** Description of desired functionality told from the perspective of the user
  - **Example:** As a registered user, I want to reset my password so that I can back into the site if I forget my password

- **Epics.** An Agile Epic is a group of related User Stories, representing a feature. You would be unlikely to introduce an Epic into a sprint without first breaking it down into it’s component User Stories so as to reduce uncertainty
  - **Example:** Allow users to create user profiles

- **Themes.** Top-level objective that may span projects and products. Themes may be broken down into sub-themes, which are more likely to be product-specific. At its most granular form, a Theme may be an Epic
  - **Example:** Personalize user experience
User stories have 3 parts

**Title** is used to reference the user story and should be kept very short, around 3-10 words. **For example:** *Passwords can be reset by users.*

**Description** is where the meat of the user story is kept. You can use the following template to write the description. **For example:** *As a [user role] I want to [goal] so I can [reason].*

**Acceptance Criteria** is used to determine when the user story has met the goal of the user.
The “INVEST” model

Characteristics of a good quality user story

<table>
<thead>
<tr>
<th><strong>Independent</strong></th>
<th>Reduced dependencies = easier to plan</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Negotiable</strong></td>
<td>Details added via collaboration</td>
</tr>
<tr>
<td><strong>Valuable</strong></td>
<td>Provides value to the customer</td>
</tr>
<tr>
<td><strong>Estimable</strong></td>
<td>Too big or too vague = not estimable</td>
</tr>
<tr>
<td><strong>Small</strong></td>
<td>Can be done in less than a week by the team</td>
</tr>
<tr>
<td><strong>Testable</strong></td>
<td>Good acceptance criteria</td>
</tr>
</tbody>
</table>
What are some of the differences between methods?

Requirements elicitation

**Waterfall-SDLC**
- Long-term planning
- Plan for creating thousands of independent shall statements which the business or system must adhere to

**Agile-SDLC**
- Small increments - Documentation is secondary to the collaboration
- Plan for soliciting user stories in initial Product Backlog
What are some of the differences between methods?

Requirements development

<table>
<thead>
<tr>
<th>Waterfall-SDLC</th>
<th>Agile-SDLC</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-level &amp; detailed reqs captured in Business &amp; Technical Req Document (BTRD)</td>
<td>User stories in Product Backlog (High-level) &amp; Iteration Backlog (Detailed)</td>
</tr>
<tr>
<td>Reqs written to limit interpretation - completeness, formality and rigidity is the rule</td>
<td>User stories written as narrative text focusing on value user gets from the system</td>
</tr>
<tr>
<td>Rarely contains explicit tests / acceptance criteria</td>
<td>Contains User Acceptance Criteria</td>
</tr>
</tbody>
</table>
What are some of the differences between methods?

Requirements management

Waterfall-SDLC

- Formal requirement reviews twice in SDLC
- Software Requirements Specification (SRS)

Agile-SDLC

- User story reviews at beginning & end of each iteration & during Release Planning
- Iteration Backlogs
Agenda

• Art of Writing Good Requirements
• Requirements Gathering Checklist
• Comparison of Methods

• **Key Takeaways & Tips**
• Q & A
Takeaways

Regardless of the methodology, experience suggests there are key factors when gathering requirements for any project:

- Focus & Clarity
- Method
- Language
- Author
- Accuracy
- Format
Takeaways

Guidelines to consider when gathering requirements

Focus & Clarity
- To see if a requirement is well defined, read it from your audience’s perspective. How does it read out loud?

Method
- Hardest part is not documenting what the customer wants but helping them figure out what they need that can also be successfully delivered.

Language
- Use the same language as your client. If the document is too technical, the client is likely to assume it’s correct because they don’t really know otherwise.

Author
- An author with a great understanding of the project domain but no experience with software development is a risky choice – and vice versa.

Accuracy
- Avoid mistakes in the first place. When issues don’t surface until the work is done, it becomes far more expensive to fix.

Format
- If the people that form the project team (client, stakeholders, designers, developers) are happy with the format, that’s half the battle won.
TIP: Ask the right questions!

Customer → Salesperson → Marketing

Coder ← Designer ← Development

Tester → System → Doing what the business needs
## 10 important questions

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. Function</strong></td>
<td>What function or process must be accomplished?</td>
<td></td>
</tr>
<tr>
<td><strong>2. Frequency</strong></td>
<td>How often will the function be used? What is the peak time and volume? Is it Cyclic vs. Ad Hoc?</td>
<td></td>
</tr>
<tr>
<td><strong>3. Information</strong></td>
<td>What data is needed and where is it?</td>
<td></td>
</tr>
<tr>
<td><strong>4. Volume</strong></td>
<td>For each output, how many pages, screens, etc.. is the user going to produce?</td>
<td></td>
</tr>
<tr>
<td><strong>5. Accuracy</strong></td>
<td>What errors can the user tolerate?</td>
<td></td>
</tr>
<tr>
<td><strong>6. Urgency</strong></td>
<td>Responsiveness: How long after should the person have to wait to receive the information? Input: How long should system wait for a user to enter the data?</td>
<td></td>
</tr>
<tr>
<td><strong>7. Natural Limits</strong></td>
<td>Any limits enforced by nature (e.g. space, size, speed)?</td>
<td></td>
</tr>
<tr>
<td><strong>8. Laws &amp; Regulations</strong></td>
<td>Any legal constraints enforced by governing agencies?</td>
<td></td>
</tr>
<tr>
<td><strong>9. Policies &amp; Rules</strong></td>
<td>Any rules imposed by business?</td>
<td></td>
</tr>
<tr>
<td><strong>10. People</strong></td>
<td>Any limitations or constraints enforced by users (e.g. ability, location, training, security, culture)</td>
<td></td>
</tr>
</tbody>
</table>
Final thoughts

- There are no silver bullets
- Measure twice, cut once
- Emphasize “what” should be done, not “how” to do it
- Avoid mistakes in the first place
Agenda

- Introduction
- Art of Writing Good Requirements
- Requirements Gathering Checklist
- Comparison of Methods
- Key Takeaways & Tips
- Q & A