

Requirements Management & Requirements Development

Robin V. Horth, PMP

Miami Valley Chapter

Project Management Institute

Introduction

- Two complementary disciplines
 - Requirements Management
 - What are requirements?
 - What is a *good* requirement?
 - What are some requirements types?
 - Requirements Development
 - Stakeholders
 - Eliciting requirement techniques
 - Reusing requirements
 - Wrap-up

Why Requirements Management & Requirements Development?

- ❑ Capability Maturity Model Integration (CMMI[®]) (An American National Standards Institute {ANSI} Standard)
- ❑ Institute of Electrical and Electronic Engineers (IEEE) Standards
- ❑ Project Management Body of Knowledge (PMBOK)
- ❑ Most of all – the quality of requirements determines the quality of your product

What is Requirements Management?

- What is a requirement?
 - A capability that a system must provide
- Requirements Management is –
 - A systematic approach to eliciting, organizing, and documenting the requirements of a system
 - A process that establishes and maintains agreement between the customer and the project team on the changing requirements of a system
- Conformance to some set of requirements defines the success of the project

What's a Good Requirement?

- Requirements
 - Correct
 - Feasible
 - Necessary
 - Prioritized
 - Unambiguous
 - Verifiable

- Requirements Specs
 - Complete
 - Consistent
 - Modifiable
 - Traceable

Why is Disciplined Requirements Management so Important?

- To satisfy the customer
 - On time
 - On budget
 - Meets needs and expectations
- To accommodate change
 - Customers needs may change during the course of development
- You must know who the real customer is...

Managing Changes to Requirements

- Requirements will change
 - Accommodating change is a measure of your team's sensitivity to stakeholders
 - Change is not the enemy – unmanaged change is
- Managing change includes
 - Keeping track of the history of each requirement
 - Configuration Management—Lock Down/Sign Off
 - Establishing traceability relationships between related requirements
 - Maintaining version control

Functional Requirements & Non-Functional Requirements

Functional Requirements

- Specific transformations of inputs to outputs

Non-Functional Requirements (Technical Requirements)

- Qualities that the product must possess
 - Security
 - Compatibility with existing systems
 - Performance requirements

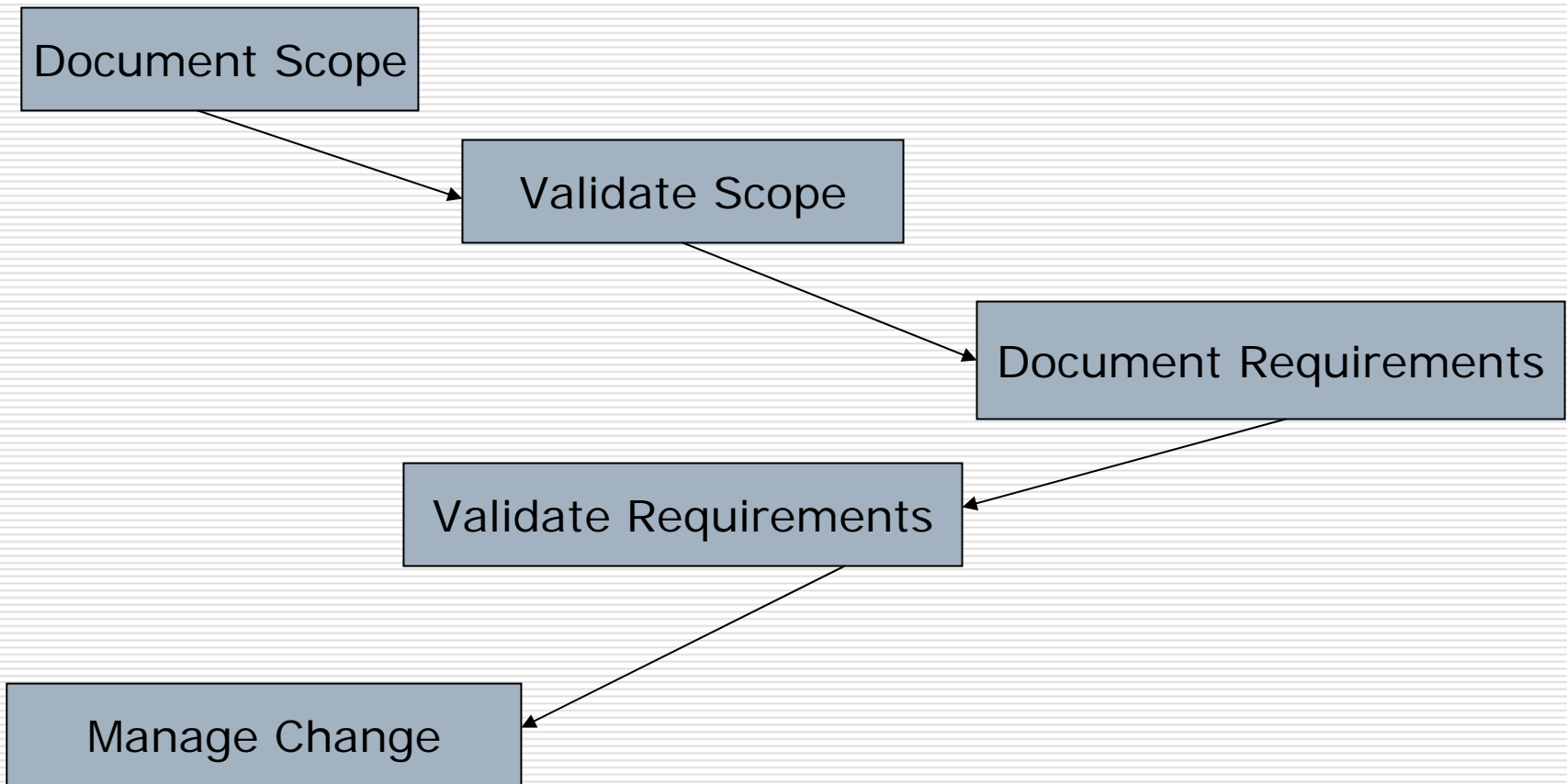
Interface Requirements

- ❑ User interfaces
- ❑ Hardware interfaces
- ❑ Software interfaces
- ❑ Communication protocols and interfaces
- ❑ **Each requirement shall be assigned a project-unique identifier to support testing and traceability and shall be stated in such a way that an objective test can be defined for it.**

Example Interface Requirements

Element	Data Type	Size	Required	Note
Code	String	50	Y	Code of the offer
Year	Integer		Y	Year of the offer
Action ID	String	100	Y	

Requirements Process



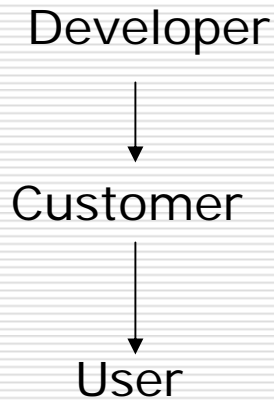
Reuse Requirements

- ❑ Saves money and time. Studies have shown that similar systems can re-use up to 80% of the requirements.
- ❑ Reuse reduces risk. Reused requirements have a better chance of being understood by all the stakeholders.
- ❑ Requirements reuse may lead to additional reuse in other lifecycle activities.
 - Component design
 - Tests
 - Code

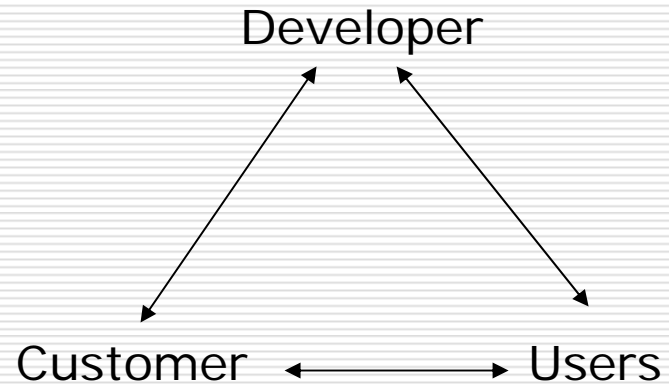
What is Requirements Development?

- The most important step in the system development process is the accurate communication of operational requirements from those who need the system to those who will build it.

Requirements Elicitation



Traditional Requirements Elicitation



Better Requirements Elicitation

Requirements Elicitation

- Problems with traditional ways of specifying problems:
 - customer may not adequately convey the needs of the user.
 - developer may not be an expert in the application domain, which inhibits communications.
 - users and customers may not understand the requirements produced by the developer
 - developer's requirements specifications typically specifies system attributes such as
 - functions,
 - performance factors,
 - design constraints,
 - system interfaces and
 - quality attributes,
 - but typically contains little or no information concerning operational characteristics of the specified system.

Requirements Elicitation Guidelines

- ❑ Assess System **Feasibility**
- ❑ Be sensitive to **organizational** and **political** considerations
- ❑ Identify and consult system **stakeholders**
- ❑ Record requirements **sources**
- ❑ Use **Business concerns** to drive requirements elicitation
- ❑ Look for **domain constraints**
- ❑ Record requirements **rationale**
- ❑ Collect requirements from **multiple viewpoints**
- ❑ **Prototype** poorly understood requirements
- ❑ Use **scenarios** to elicit requirements
- ❑ Define **operational processes**
- ❑ **Reuse** requirements

Requirements Elicitation Techniques

- Interviewing and questionnaires
- Requirements workshops
- Braining Storming and idea reduction
- Storyboards
- Use Cases
- Role Playing
- Prototyping

Identify and Consult System Stakeholders

- If lacking consideration of **everyone who is likely to be affected** by the introduction of the system, there is a great likelihood of missing some critical requirements.
- "Identifying stakeholders and discussing the system with them makes people feel like they are part of the requirements elicitation process. In fact, it *makes* them a part of it."

Summary

- Two complementary disciplines
 - Requirements Management
 - What are requirements?
 - What is a *good* requirement?
 - What are some requirements types?
 - Requirements Development
 - Stakeholders
 - Eliciting requirement techniques
 - Reusing requirements
 - Where can you get more information?

References

- ❑ 1 "Requirements Engineering A good practice guide," Ian Sommerville and Pete Sawyer, John Wiley and Sons, 1997
- ❑ 2 "Managing Software Requirements; A Unified Approach," Dean Leffingwell and Don Widrig, Addison-Wesley, 2000
- ❑ 3 Software Quality Measurement for Distributed Systems, RADC-TR-83-175
- ❑ 4 Requirements Engineering, Thayer, SMC 10/97, version 2
- ❑ 5 Richard Thayer, *Software Requirements Engineering*, IEEE, 1997
- ❑ 6 STEP, Operational Requirements for Automated Capabilities, STEP, 1991
- ❑ 7 MBASE, "Avoiding the Software Model-Clash Spiderweb," IEEE Computer, November, 2000, pp. 120-122.

References

- ❑ 8 RequisitePro Tutorial
- ❑ 9 Karl Wiegers – Process Impact.com
- ❑ 10 Wikipedia.org
- ❑ 11 R. E. Fairley and R. H. Thayer, "The Concept of Operations: The Bridge from Operational Requirements to Technical Specifications," *Software Engineering*, M. Dorfman and R. H. Thayer (eds.), IEEE Comp. Soc. Press, Los Alamitos, CA, 1997.
- ❑ 12 Michael Madigan, StorageTek
- ❑ DoD Extensions to the PMBOK
- ❑ INCOSE, George Orr, PhD