# Negotiation & Prioritization

TOPIC # 9
Chapter 15, 16 & 17– Karl Wiegers
Chapter 11,13,14 - Reference

## Agreeing on a Specification

- □ Two key problems for getting agreement:
  - 1) the problem of validation
    - *If we build to this spec, will the customer's expectations be met?
  - 2) the problem of negotiation
    - *How do you reconcile conflicting goals in a complex socio-cognitive setting?
- □ Validating Requirements
  - □ Inspections and Reviews
  - □ Prototyping
- □ Negotiating Requirements
  - □ Requirements Prioritization
  - □ Conflict and Conflict Resolution
  - □ Requirements Negotiation Techniques

## Prototyping

- Software Prototyping puts a mock-up or an initial slice of a new system in front of users to stimulate their thinking & catalyze the requirements dialogs.
- Reduces the risk of customer dissatisfaction by early feedback
- "I will know it when I see it" – IKIWISI
- Clarify, complete and validate requirements
- Explore design alternatives
- Create a subset that will grow into the ultimate product
- Resolve uncertainties in the early development cycle
- Paper (low fidelty) V/s Electronic Prototype (high fidelty)

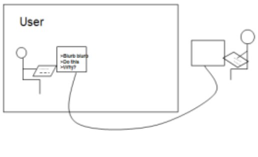
## Prototyping

- □ Definitions
  - □ "A software prototype is a partial implementation constructed primarily to enable customers, users, or developers to learn more about a problem or its solution." [Davis 1990]
  - □ "Prototyping is the process of building a working model of the system" [Agresti 1986]
- □ Approaches to prototyping
  - □ Presentation Prototypes
    - □ explain, demonstrate and inform - then throw away
    - □ e.g. used for proof of concept; explaining design features; etc.
  - □ Exploratory Prototypes
    - □ used to determine problems, elicit needs, clarify goals, compare design options
    - □ informal, unstructured and thrown away.
  - □ Breadboards or Experimental Prototypes
    - □ explore technical feasibility; test suitability of a technology
    - □ Typically no user/customer involvement
  - □ Evolutionary (e.g. "operational prototypes", "pilot systems"):
    - □ development seen as continuous process of adapting the system
    - □ "prototype" is an early deliverable, to be continually improved.

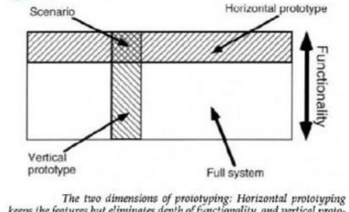
## Wizard of Oz Prototyping

❑ The user thinks they are interacting with a computer, but a developer is responding to output rather than the system.

❑ Usually done early in design to understand users' expectations



## Horizontal v/s vertical prototyping



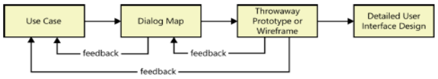
*The two dimensions of prototyping: Horizontal prototyping keeps the features but eliminates depth of functionality, and vertical prototyping gives full functionality for a few features.*

# Throwaway or Evolve?

- Throwaway Prototyping
  - Purpose:
    - to learn more about the problem or its solution…
    - hence discard after the desired knowledge is gained.
  - Use:
    - early or late
  - Approach:
    - horizontal - build only one layer (e.g. UI)
    - “quick and dirty”
  - Advantages:
    - Learning medium for better convergence
    - Early delivery → early testing → less cost
    - Successful even if it fails!
  - Disadvantages:
    - Wasted effort if requirements change rapidly
    - Often replaces proper documentation of the requirements
    - May set customers’ expectations too high
    - Can get developed into final product
- Evolutionary Prototyping
  - Purpose
    - to learn more about the problem or its solution…
    - …and to reduce risk by building parts of the system early
  - Use:
    - incremental; evolutionary
  - Approach:
    - vertical - partial implementation of all layers;
    - designed to be extended/adapted
  - Advantages:
    - Requirements not frozen
    - Return to last increment if error is found
    - Flexible(?)
  - Disadvantages:
    - Can end up with complex, unstructured system which is hard to maintain
    - early architectural choice may be poor
    - Optimal solutions not guaranteed
    - Lacks control and direction

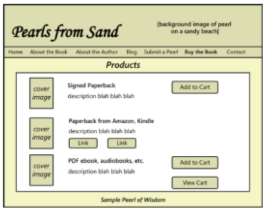
*Brooks: “Plan to throw one away - you will anyway!”

# Working with prototypes

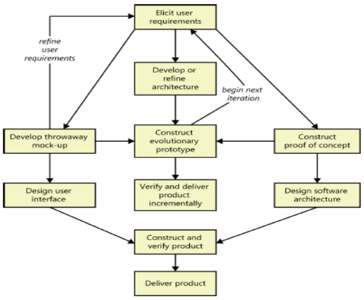

Use Case
Dialog Map
Throwaway Prototype or Wireframe
Detailed User Interface Design
feedback
feedback
feedback

Activity sequence from use cases to user interface design using a throwaway prototype.

- A throwaway prototype or a wireframe elaborates the dialog elements into specific screens, menus, and dialog boxes.



# Ways for prototyping in s/w development process

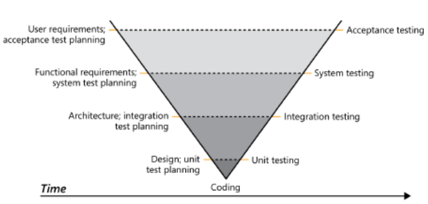

Elicit user requirements
refine user requirements
Develop or refine architecture
begin next iteration
Develop throwaway mock-up
Construct evolutionary prototype
Construct proof of concept
Design user interface
Verify and deliver product incrementally
Design software architecture
Construct and verify product
Deliver product

Several possible ways to incorporate prototyping into the software development process.

# Risks of Prototyping

- Pressure to release prototype
- Distraction by details
- Unrealistic Performance Expectation
- Investing excessive efforts in prototypes

# Validating the requirements


User requirements; acceptance test planning
Acceptance testing
Functional requirements; system test planning
System testing
Architecture; integration test planning
Integration testing
Design; unit test planning
Unit testing
Coding
Time

The V model of software development incorporates early test planning and test design.

# Validation & Verification purpose

- The software requirements accurately describe the intended system capabilities and properties that will satisfy the various stakeholders’ needs.
- The software requirements are correctly derived from the business requirements, system requirements, business rules, and other sources.
- The requirements are complete, feasible, and verifiable.
- All requirements are necessary, and the entire set is sufficient to meet the business objectives.
- All requirements representations are consistent with each other.
- The requirements provide an adequate basis to proceed with design and construction.

# Reviewing Requirements

- A *peer deskcheck,* in which you ask one colleague to look over your work product.
- A *passaround,* in which you invite several colleagues to examine a deliverable concurrently.
- A *walkthrough,* during which the author describes a deliverable and solicits comments on it.

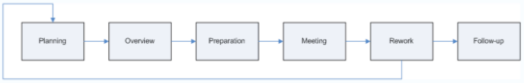
# Inspection Roles

- Roles
  - Author
    - The person who created the work product being inspected
  - Moderator
    - This is the leader of the inspection. The moderator plans the inspection and coordinates it
  - Reader
    - The person reading through the documents, one item at a time. The other inspectors then point out defects
  - Recorder/Scribe
    - The person that documents the defects that are found during the inspection
  - Inspector
    - The person that examines the work product to identify possible detects

# Inspection Process

Process

- Planning
  - The inspection is planned by the moderator
- Overview meeting
  - The author describes the background of the work product
- Preparation
  - Each inspector examines the work product to identify possible defects
- Inspection meeting
  - During this meeting, the reader reads through the work product, part by part and the inspectors point out the defects for every part
- (Rework
  - The author makes change to the work product according to the action plans from the inspection meeting
- Follow-up
  - The changes by the author are checked to make sure everything is correct)



# Inspection Constraints

Source: Adapted from Blum, 1992, pp369-373 & Freedman and Weinberg, 1990.

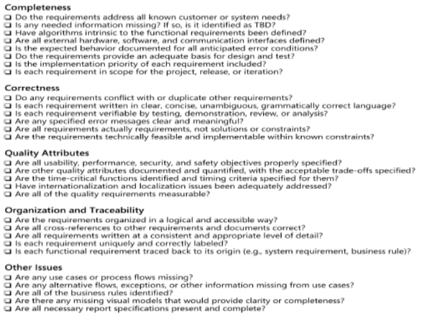
- Size
  - "enough people so that all the relevant expertise is available"
  - min: 3 (4 if author is present)
  - max: 7 (less if leader is inexperienced)
- Duration
  - never more than 2 hours
    - concentration will flag if longer
- Outputs
  - all reviewers must agree on the result
    - accept; re-work; re-inspect;
  - all findings should be documented
    - summary report (for management)
    - detailed list of issues
- Scope
  - focus on small part of a design, not the whole thing
- Timing
  - Examines a product once its author has finished it
  - not too soon
    - product not ready - find problems the author is already aware of
  - not too late
    - product in use - errors are now very costly to fix
- Purpose
  - Remember the biggest gains come from fixing the process
    - collect data to help you not to make the same errors next time

# Inspection Guidelines

Source: Adapted from Freedman and Weinberg, 1990.

- Prior to the review
  - schedule Formal Reviews into the project planning
  - train all reviewers
  - ensure all attendees prepare in advance
- During the review
  - review the product, not its author
    - keep comments constructive, professional and task-focussed
  - stick to the agenda
    - leader must prevent drift
  - limit debate and rebuttal
    - record issues for later discussion/resolution
  - identify problems but don't try to solve them
  - take written notes
- After the review
  - review the review process

# Sample inspection checklist for SRS



**Completeness**

❑ Do the requirements address all known customer or system needs?
❑ Is any needed information missing? If so, is it identified as TBD?
❑ Have algorithms intrinsic to the functional requirements been defined?
❑ Are all external hardware, software, and communication interfaces defined?
❑ Is the expected behavior documented for all anticipated error conditions?
❑ Do the requirements provide an adequate basis for design and test?
❑ Is the implementation priority of each requirement included?
❑ Is each requirement in scope for the project, release, or iteration?

**Correctness**

❑ Do any requirements conflict with or duplicate other requirements?
❑ Is each requirement written in clear, concise, unambiguous, grammatically correct language?
❑ Is each requirement verifiable by testing, demonstration, review, or analysis?
❑ Are any specified error messages clear and meaningful?
❑ Are all requirements actually requirements, not solutions or constraints?
❑ Are the requirements technically feasible and implementable within known constraints?

**Quality Attributes**

❑ Are all usability, performance, security, and safety objectives properly specified?
❑ Are other quality attributes documented and quantified, with the acceptable trade-offs specified?
❑ Are the time-critical functions identified and timing criteria specified for them?
❑ Have internationalization and localization issues been adequately addressed?
❑ Are all of the quality requirements measurable?

**Organization and Traceability**

❑ Are the requirements organized in a logical and accessible way?
❑ Are all cross-references to other requirements and documents correct?
❑ Are all requirements written at a consistent and appropriate level of detail?
❑ Is each requirement uniquely and correctly labeled?
❑ Is each functional requirement traced back to its origin (e.g., system requirement, business rule)?

**Other Issues**

❑ Are any use cases or process flows missing?
❑ Are any alternative flows, exceptions, or other information missing from use cases?
❑ Are all of the business rules identified?
❑ Are there any missing visual models that would provide clarity or completeness?
❑ Are all necessary report specifications present and complete?

### Inspection ways to review requirement document

- **Adhoc Review:** A review with no formal, systematic procedure, based only individual experience.
- **Checklist Review:** A list of items is provided to reviewers, which makes this inspection process more focused

## Types of Requirement Errors

- **Error of omission/oversight:** Domain experts easily forget to convey domain knowledge to requirements engineers, because they consider that to be obvious and implicit
- **Error of ambiguity and clarity:** Use of ambiguous and unclear language .. Using English language to document requirements
- **Error of commission:** Including something that was not important or not required.
- **Performance Errors:** Due to conflicting need and understanding of stakeholders, some quality requirements which are not necessary might be given privilege to other

## Requirements Negotiation

- Possible conflicts to be resolved among stakeholders
  - Between supplier and customers about costs, benefits, risks
  - Power struggle within customer organization
  - Conflicts with other projects about resources
  - Conflicting goals, features, requirements
- Conflict resolution involves negotiation
  - Negotiating a coherent set of requirements is not easy
  - But it is one task of the requirements analyst
  - Difficult to satisfy everyone, to achieve all goals, make good decisions!
  - Involves a lot of (group) discussions

21

## Requirement Negotiation

- Negotiations are rarely conducted using only logical and technical arguments
- They are influenced by organizational and political considerations, and the personalities of the people involved
- A strong personality may force their priorities on other stakeholders
- Requirements may be accepted or rejected because they strengthen the political influence in the organization of some stakeholders
- **Meetings** are the most effective way to negotiate requirements are resolve requirement conflicts

## Types of politics in RE

- Two types:
  - – *Functional politics* – which of problems deserve attention, whose interests will be served – concerns about the functional intent of the system.
  - – *Resource politics* – to solving which of problems to allocate the available resources – concerns about the flow of resources going into the system.
- Resource politics is often hidden, as proponents of a new system deliberately downplay the expected costs in order to improve the chances that the system will be funded.

## Decision models

**Common decision models**:

- *Autocracy* – an individual or close-knit group decides the political question and all others comply.
  - Is often assumed, but real autocracy is relatively rare.
- *Pluralism* - the diverse interests of various stakeholders with standing are considered on their individual merits, and efforts are made to accommodate all reasonable interests in the solution (weights of interests are not necessarily equal).
  - Negotiation can be very difficult
- *Two-party contest* - diverse interests come together behind either a proponent or opponent position on a particular set of requirements.

## Requirement Negotiation Process

- Stages of Negotiation Meeting
  - Information Stage
  - Discussion Stage
  - Resolution Stage
- **Information Stage:** An information stage where the nature of the problems associated with a requirement is explained
- **Discussion Stage:** A discussion stage where the stakeholders involved discuss how these problems might be resolved
  - All stakeholders with an interest in the requirement should be given the opportunity to comment. Priorities may be assigned to requirements at this stage

## Requirement Negotiation Process

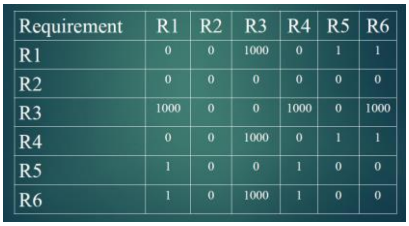
- **Resolution Stage:**
- A resolution stage where actions concerning the requirement are agreed
  - These actions might be to delete the requirement, to suggest specific modifications to the requirement or to elicit further information about the requirement

## Requirement Interactions & Conflicts

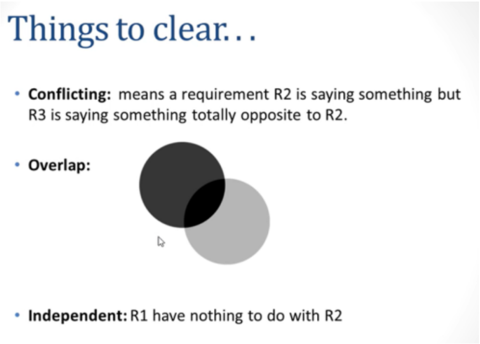
- A very important objective of requirements analysis is to discover the interactions between requirements and to highlight requirements **conflicts and overlaps**
- A requirements interaction matrix shows how requirements interact with each other, which can be constructed using a spreadsheet
- Each requirement is compared with other requirements, and the matrix is filled as follows:
  - For requirements which conflict, fill in a 1
  - For requirements which overlap, fill in a 1000
  - For requirements which are independent, fill in a 0
  - If you can't decide whether requirements conflict, you should assume that a conflict exists.

## Requirement Interaction Matrix

- The advantage of using numeric values for conflicts and overlaps is that you can sum each row and column to find the number of conflicts and the number of overlaps



| Requirement | R1 | R2 | R3 | R4 | R5 | R6 |
|---|---|---|---|---|---|---|
| R1 | 0 | 0 | 1000 | 0 | 1 | 1 |
| R2 | 0 | 0 | 0 | 0 | 0 | 0 |
| R3 | 1000 | 0 | 0 | 1000 | 0 | 1000 |
| R4 | 0 | 0 | 1000 | 0 | 1 | 1 |
| R5 | 1 | 0 | 0 | 1 | 0 | 0 |
| R6 | 1 | 0 | 1000 | 1 | 0 | 0 |



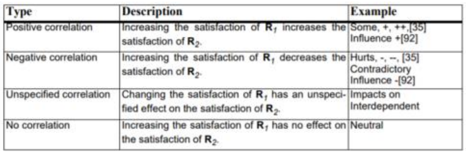
## Things to clear…

- **Conflicting:** means a requirement R2 is saying something but R3 is saying something totally opposite to R2.
- **Overlap:**
- **Independent:** R1 have nothing to do with R2

## Problems that lead to requirements conflicts.

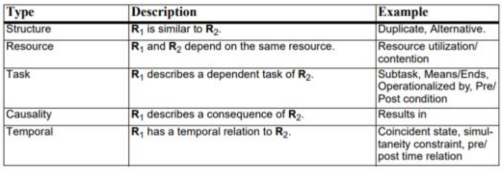
- Voluminous requirements.
- Changing requirements and analysts.
- Complex requirements.
- Conflicting stakeholder requirements.
- Changing and unidentified stakeholders.
- Changing expectations.

## Types of Requirement Relationships



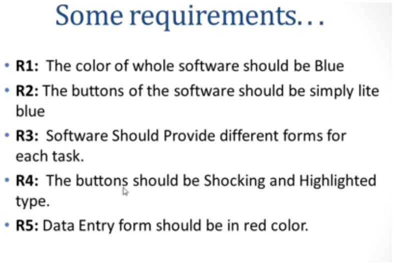
| Type | Description | Example |
|---|---|---|
| Positive correlation | Increasing the satisfaction of **$R_1$** increases the satisfaction of **$R_2$**. | Some, +, ++,[35]<br>Influence +[92] |
| Negative correlation | Increasing the satisfaction of **$R_1$** decreases the satisfaction of **$R_2$**. | Hurts, -, --, [35]<br>Contradictory Influence -[92] |
| Unspecified correlation | Changing the satisfaction of **$R_1$** has an unspecified effect on the satisfaction of **$R_2$**. | Impacts on<br>Interdependent |
| No correlation | Increasing the satisfaction of **$R_1$** has no effect on the satisfaction of **$R_2$**. | Neutral |

## Basis of Requirement Relationships



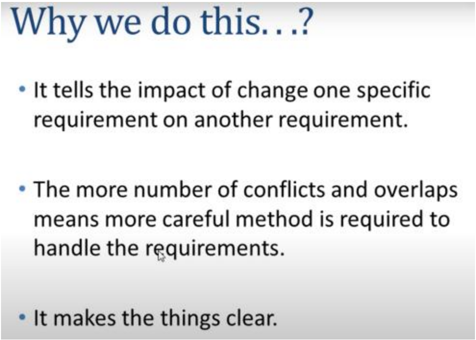
| Type | Description | Example |
|---|---|---|
| Structure | **$R_1$** is similar to **$R_2$**. | Duplicate, Alternative. |
| Resource | **$R_1$** and **$R_2$** depend on the same resource. | Resource utilization/ contention |
| Task | **$R_1$** describes a dependent task of **$R_2$**. | Subtask, Means/Ends, Operationalized by, Pre/ Post condition |
| Causality | **$R_1$** describes a consequence of **$R_2$**. | Results in |
| Temporal | **$R_1$** has a temporal relation to **$R_2$**. | Coincident state, simultaneity constraint, pre/ post time relation |

## Example



Some requirements…

- **R1:** The color of whole software should be Blue
- **R2:** The buttons of the software should be simply lite blue
- **R3:** Software Should Provide different forms for each task.
- **R4:** The buttons should be Shocking and Highlighted type.
- **R5:** Data Entry form should be in red color.



Why we do this…?

- It tells the impact of change one specific requirement on another requirement.
- The more number of conflicts and overlaps means more careful method is required to handle the requirements.
- It makes the things clear.

## Requirements Prioritization

- Prioritization requires an understanding of six issues
  - The needs of the customers
  - The relative importance of requirements to the customers
  - The timing at which capabilities need to be delivered
  - Requirements that serve as predecessors for other requirements and other relationships among requirements
  - Which requirements must be implemented as a group
  - The cost to satisfy each requirement

## Prioritization Techniques

- On a small project, the stakeholders should be able to agree on requirement priorities informally.
- **In or out**: The simplest of all prioritization methods is to have a group of stakeholders work down a list of requirements and make a binary decision: is it in, or is it out?
- **Pairwise comparison & rank ordering:** People sometimes try to assign a unique priority sequence number to each requirement. Rank ordering a list of requirements involves making pairwise comparisons between all of them so you can judge which member of each pair has higher priority.

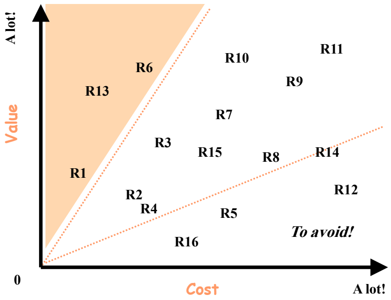
## Requirements Analysis

- **Benefit/penalty/cost/risk analysis**. Rate for each alternative or requirement:
  - The relative benefit of the requirement, e.g. from 1 (useless) to 9 (extremely valuable).
  - The relative penalty that will be suffered if the requirement is not included, e.g. from 1 (no one is upset) to 9 (very serious consequences).
  - The relative cost of implementing the requirement, e.g. from 1 (quick and easy) to 9 (very time consuming and expensive).
  - The relative risk of not getting the requirement implemented right, e.g. form 1 (no risk) to 9 (significant risks exist related to available expertise, reliability of tools, technology or partners).
- **Interaction analysis**. Analyze interaction between different requirements, what impact on other parts of the system will be when a particular alternative is selected, etc.

## 80-20 Rule

- 20% of functionalities provide 80% of revenues
  - Think of MS Word…
- The remaining 80% of functionalities offer a lower return on investment while adding delays, development costs, maintenance costs…
- How to find the most useful and beneficial 20% of functionalities?

## Which Sector Should We Focus On?



## Technique – Prioritization Scales

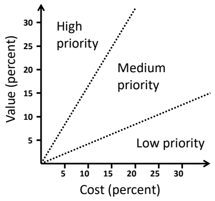
- Determine criteria, granularity, scale dimensions
- Frequently used:
  - Urgency
    - High (mission critical requirement; required for next release)
    - Medium (supports necessary system operations; required eventually but could wait until a later release if necessary)
    - Low (a functional or quality enhancement; would be nice to have someday if resources permit)
  - Importance
    - Essential (product unacceptable unless these requirements are satisfied)
    - Conditional (would enhance the product, but the product is acceptable if absent)
    - Optional (functions that ma or may not be worthwhile)

| | Important | Not Important |
|---|---|---|
| Urgent | High Priority | Low Priority |
| Not Urgent | Medium Priority | Don't do these |

## A Cost-Value Approach

Calculate return on investment

- Assess each requirement's importance to the project as a whole
- Assess the relative cost of each requirement
- Compute the cost-value trade-off:



## Estimating Cost & Value

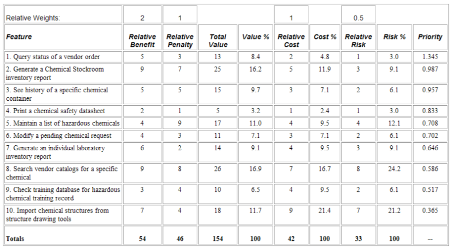
- Two approaches:
  - Absolute scale (e.g. dollar values)
    - Requires much domain experience
  - Relative values (e.g. less/more; a little, somewhat, very)
    - Much easier to elicit
    - Prioritization becomes a sorting problem

## Technique – Wiegers' Prioritization

- Semi-quantitative analytical approach to requirements prioritization based on value, cost, and risk
- Relies on estimation of relative priorities of requirements
  - Dimensions
    - Relative benefit (for having requirement)
    - Relative penalty to stakeholder (if requirement is not included)
    - Relative cost (to implement requirement)
    - Relative risk (technical and other risks)
  - Each dimension is given a value on a given scale (e.g., 0..9)
  - Dimensions have relative weights
- Formula used to derive overall priority
  - priority = (value%) / ((cost% * cost weight) + (risk% * risk weight))
- Still limited by ability to properly estimate
  - Requires adaptation and calibration

## Wiegers' Prioritization Example

Chemical tracking system



| Relative Weights: | 2 | 1 | | | 1 | | 0.5 | | |
|---|---|---|---|---|---|---|---|---|---|
| ***Feature*** | ***Relative Benefit*** | ***Relative Penalty*** | ***Total Value*** | ***Value %*** | ***Relative Cost*** | ***Cost %*** | ***Relative Risk*** | ***Risk %*** | ***Priority*** |
| 1. Query status of a vendor order | 5 | 3 | 13 | 8.4 | 2 | 4.8 | 1 | 3.0 | 1.345 |
| 2. Generate a Chemical Stockroom inventory report | 9 | 7 | 25 | 16.2 | 5 | 11.9 | 3 | 9.1 | 0.987 |
| 3. See history of a specific chemical container | 5 | 5 | 15 | 9.7 | 3 | 7.1 | 2 | 6.1 | 0.957 |
| 4. Print a chemical safety datasheet | 2 | 1 | 5 | 3.2 | 1 | 2.4 | 1 | 3.0 | 0.833 |
| 5. Maintain a list of hazardous chemicals | 4 | 9 | 17 | 11.0 | 4 | 9.5 | 4 | 12.1 | 0.708 |
| 6. Modify a pending chemical request | 4 | 3 | 11 | 7.1 | 3 | 7.1 | 2 | 6.1 | 0.702 |
| 7. Generate an individual laboratory inventory report | 6 | 2 | 14 | 9.1 | 4 | 9.5 | 3 | 9.1 | 0.646 |
| 8. Search vendor catalogs for a specific chemical | 9 | 8 | 26 | 16.9 | 7 | 16.7 | 8 | 24.2 | 0.586 |
| 9. Check training database for hazardous chemical training record | 3 | 4 | 10 | 6.5 | 4 | 9.5 | 2 | 6.1 | 0.517 |
| 10. Import chemical structures from structure drawing tools | 7 | 4 | 18 | 11.7 | 9 | 21.4 | 7 | 21.2 | 0.365 |
| **Totals** | **54** | **46** | **154** | **100** | **42** | **100** | **33** | **100** | **--** |

# END OF TOPIC # 9

**-COMING UP!!!!!!**

**-Stakeholder Analysis**

**-Non Functional Requirements**

**-Risk Management & Risks**