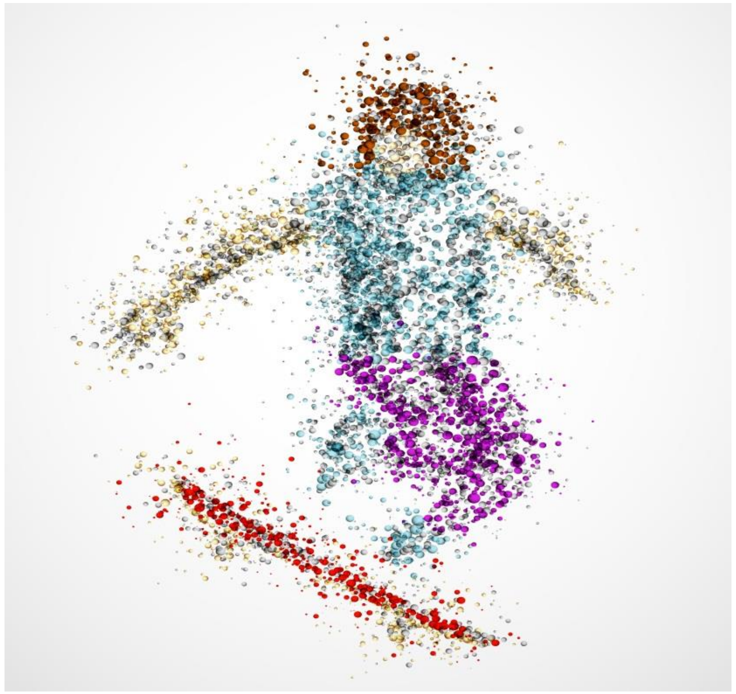


Use Cases and User Stories for Agile Requirements

Peter Schmidt, PMP, PMI-ACP, CPL
VP, Client Services, ESI International
pschmidt@esi-intl.com

www.esi-intl.com

Agenda

1

Requirements Principles

Identify the principles that lead to effective Agile requirements

2

Setting the Stage for Requirements

Establish the vision as the foundation of Agile requirements

3

Levels of Agile Requirements

Identify the different level of Agile requirements for effective requirements

Agile Requirements Principles

- Design a process for collaborative requirements gathering upfront
- Identify and engage a product owner and knowledgeable SMEs
- Acquire effective facilitation/elicitation and visual modeling skills
- Focus on breadth early, on depth later
- Break down/slice requirements to the right level
- Define ‘Acceptance Tests’ as part of the requirement
- Keep a ‘Just Enough for the Next Step’ attitude

Requirements Principles

Top 10 Agile Requirements Characteristics

1. Just in-time detail
2. Light
3. Assumes change
4. Never sealed
5. Estimated in points
6. Prioritized top down
7. Shorter time to gather
8. ONE product owner
9. Focus on breadth
10. Gathered collaboratively

Please Take the Poll

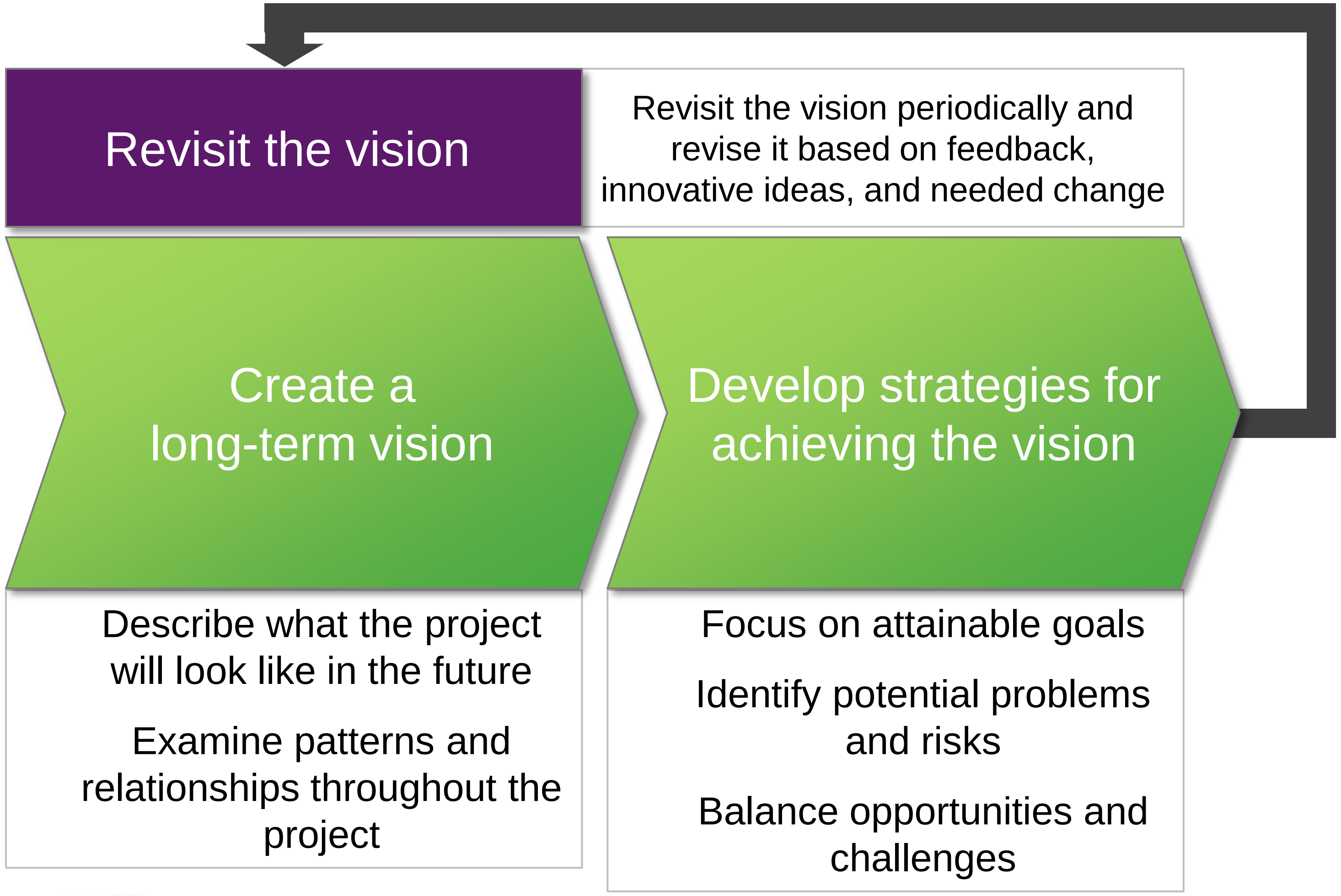
- What are the challenges you face when it comes to writing good requirements?

  - Think breadth then depth
  - Break it down into small chunks
  - Collaboration
  - Following INVEST (Independent, Negotiable, Valuable, Estimate-able, Small, Testable)

Writing Good Requirements

- Should be done collaboratively
- Think breadth then depth
- Follow the U-INVEST attributes
- Break it down into small chunks
- Stories are not 'Tasks'

Setting Project Direction



Revisit the vision

Revisit the vision periodically and revise it based on feedback, innovative ideas, and needed change

Create a long-term vision

Develop strategies for achieving the vision

Describe what the project will look like in the future

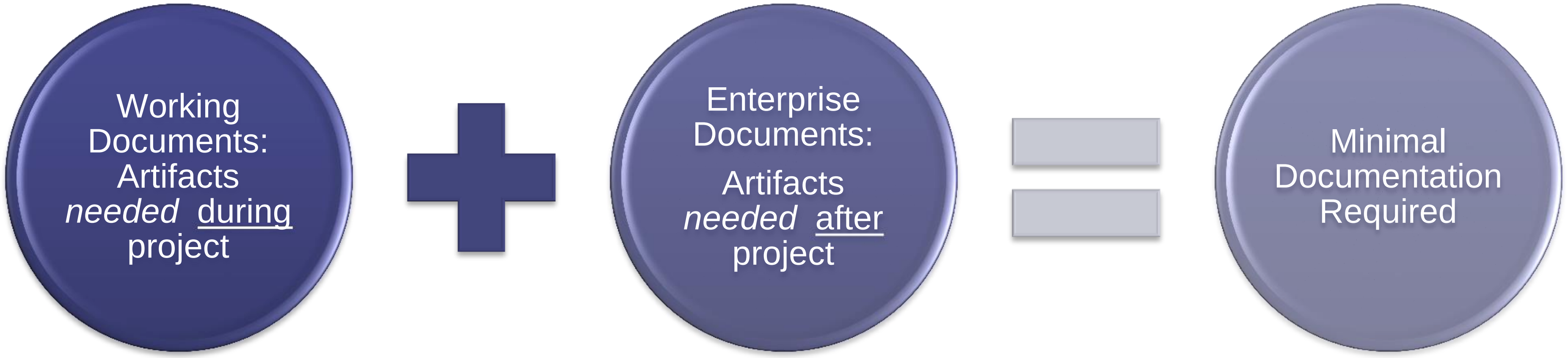
Examine patterns and relationships throughout the project

Focus on attainable goals

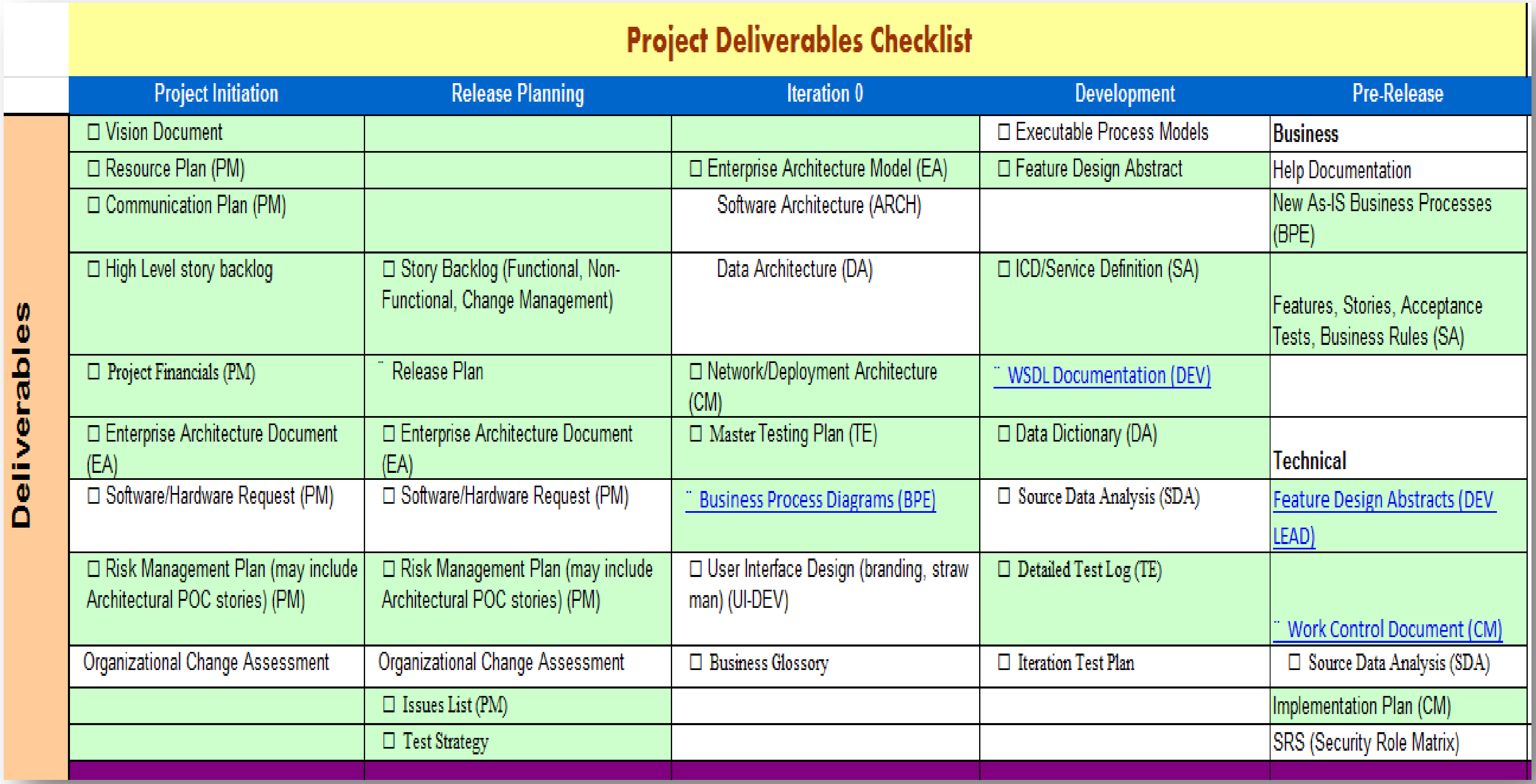
Identify potential problems and risks

Balance opportunities and challenges

Agreeing on Project Deliverables


Working Documents: Artifacts *needed* during project
+
Enterprise Documents: Artifacts *needed* after project
=
Minimal Documentation Required

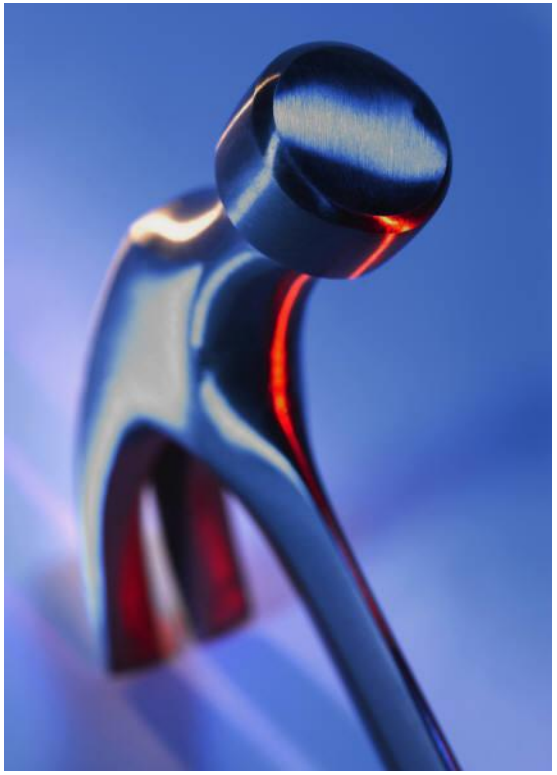
Agreeing on Project Deliverables



Project Deliverables Checklist

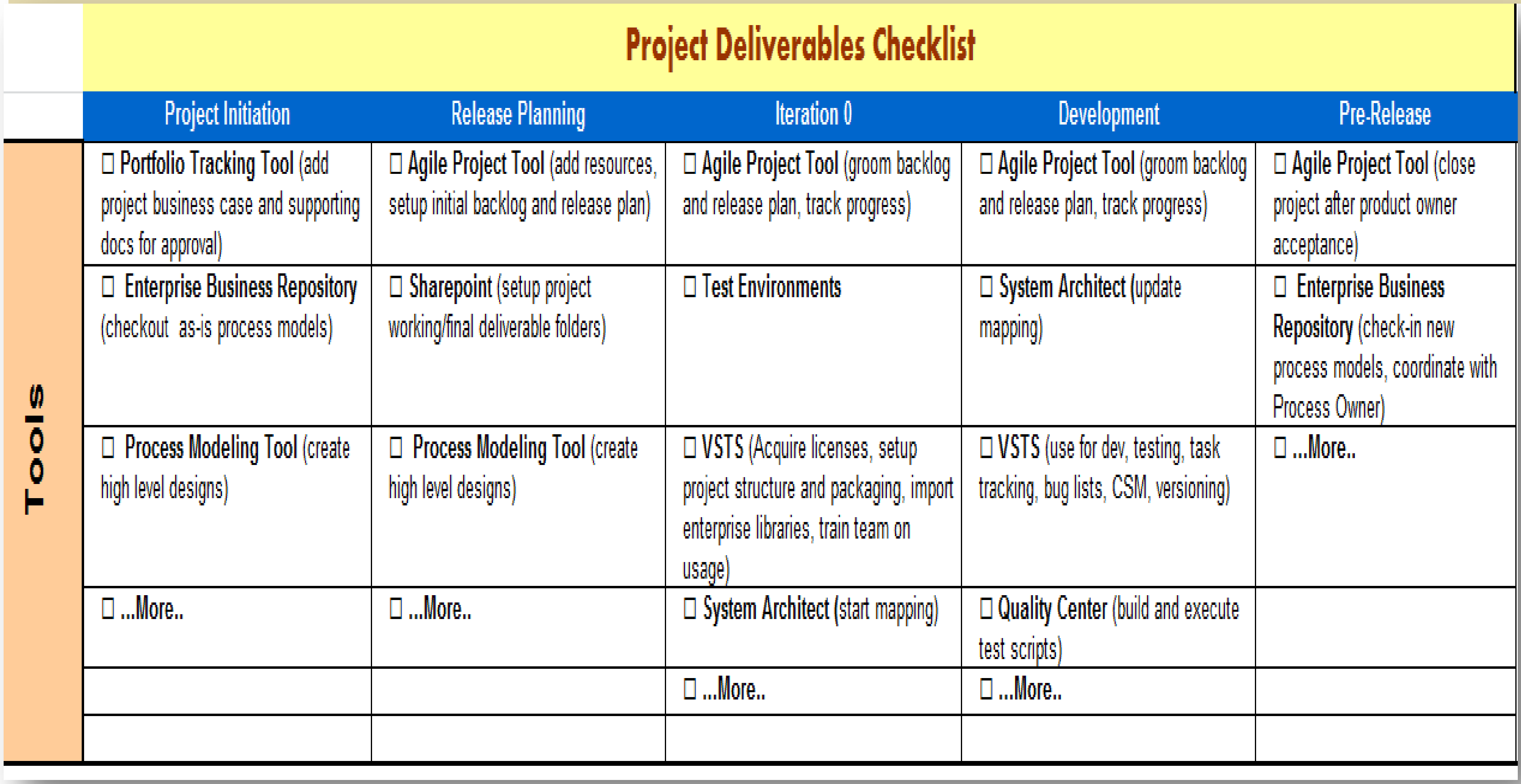
	Project Initiation	Release Planning	Iteration 0	Development	Pre-Release
Deliverables	☐ Vision Document			☐ Executable Process Models	**Business**
	☐ Resource Plan (PM)		☐ Enterprise Architecture Model (EA)	☐ Feature Design Abstract	Help Documentation
	☐ Communication Plan (PM)		Software Architecture (ARCH)		New As-IS Business Processes (BPE)
	☐ High Level story backlog	☐ Story Backlog (Functional, Non-Functional, Change Management)	Data Architecture (DA)	☐ ICD/Service Definition (SA)	Features, Stories, Acceptance Tests, Business Rules (SA)
	☐ Project Financials (PM)	¨ Release Plan	☐ Network/Deployment Architecture (CM)	¨ WSDL Documentation (DEV)	
	☐ Enterprise Architecture Document (EA)	☐ Enterprise Architecture Document (EA)	☐ Master Testing Plan (TE)	☐ Data Dictionary (DA)	**Technical**
	☐ Software/Hardware Request (PM)	☐ Software/Hardware Request (PM)	¨ Business Process Diagrams (BPE)	☐ Source Data Analysis (SDA)	Feature Design Abstracts (DEV LEAD)
	☐ Risk Management Plan (may include Architectural POC stories) (PM)	☐ Risk Management Plan (may include Architectural POC stories) (PM)	☐ User Interface Design (branding, straw man) (UI-DEV)	☐ Detailed Test Log (TE)	¨ Work Control Document (CM)
	Organizational Change Assessment	Organizational Change Assessment	☐ Business Glossory	☐ Iteration Test Plan	☐ Source Data Analysis (SDA)
		☐ Issues List (PM)			Implementation Plan (CM)
		☐ Test Strategy			SRS (Security Role Matrix)

Agreeing on Tools



- What functionality do we need? When?
- Which tools can help us reach our goal?
- What is our criteria for selecting

Agreeing on Tools

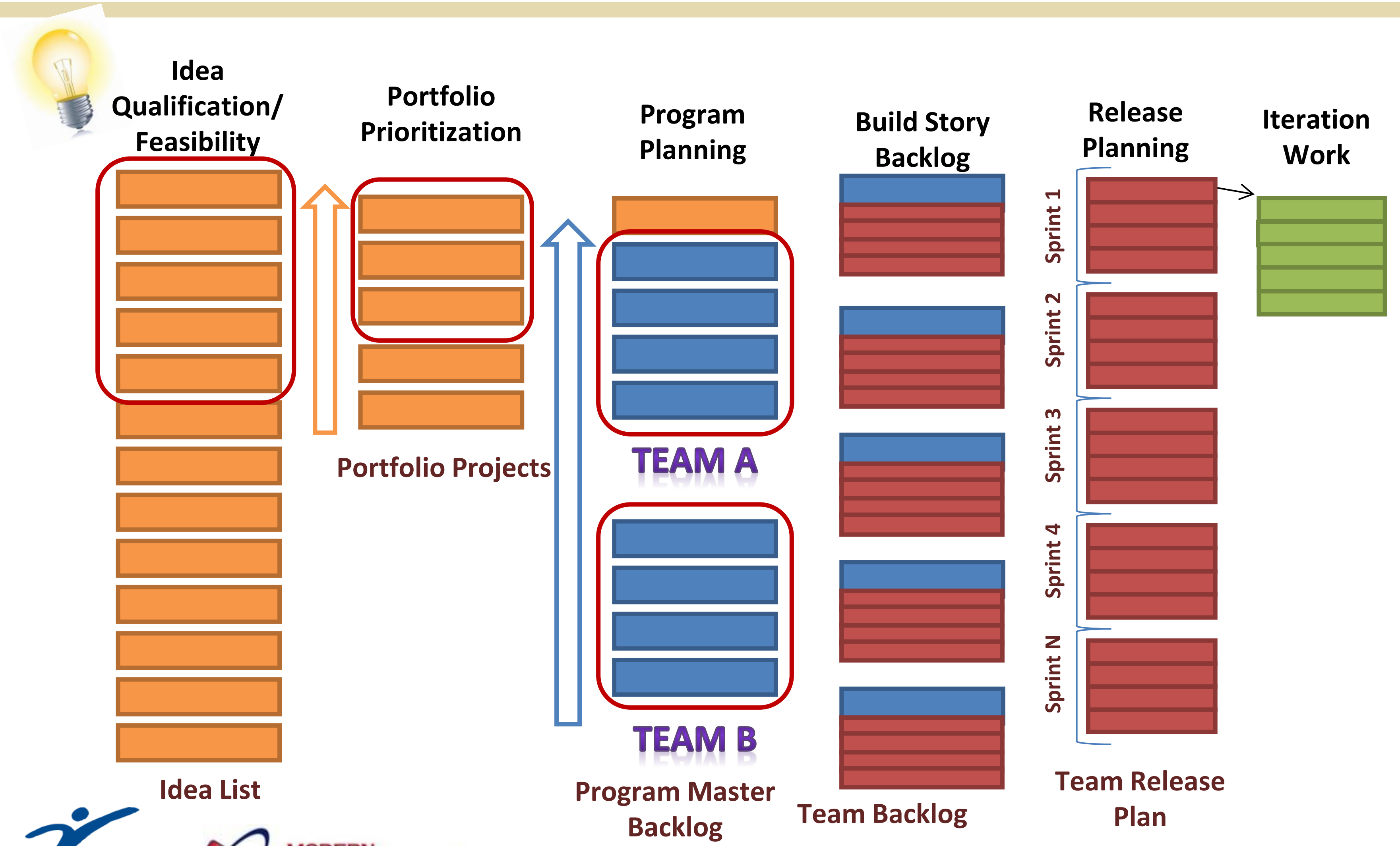


Project Deliverables Checklist

	Project Initiation	Release Planning	Iteration 0	Development	Pre-Release
Tools	☐ **Portfolio Tracking Tool** (add project business case and supporting docs for approval)	☐ **Agile Project Tool** (add resources, setup initial backlog and release plan)	☐ **Agile Project Tool** (groom backlog and release plan, track progress)	☐ **Agile Project Tool** (groom backlog and release plan, track progress)	☐ **Agile Project Tool** (close project after product owner acceptance)
	☐ **Enterprise Business Repository** (checkout as-is process models)	☐ **Sharepoint** (setup project working/final deliverable folders)	☐ **Test Environments**	☐ **System Architect** (update mapping)	☐ **Enterprise Business Repository** (check-in new process models, coordinate with Process Owner)
	☐ **Process Modeling Tool** (create high level designs)	☐ **Process Modeling Tool** (create high level designs)	☐ **VSTS** (Acquire licenses, setup project structure and packaging, import enterprise libraries, train team on usage)	☐ **VSTS** (use for dev, testing, task tracking, bug lists, CSM, versioning)	☐ **...More..**
	☐ **...More..**	☐ **...More..**	☐ **System Architect** (start mapping)	☐ **Quality Center** (build and execute test scripts)	
			☐ **...More..**	☐ **...More..**	

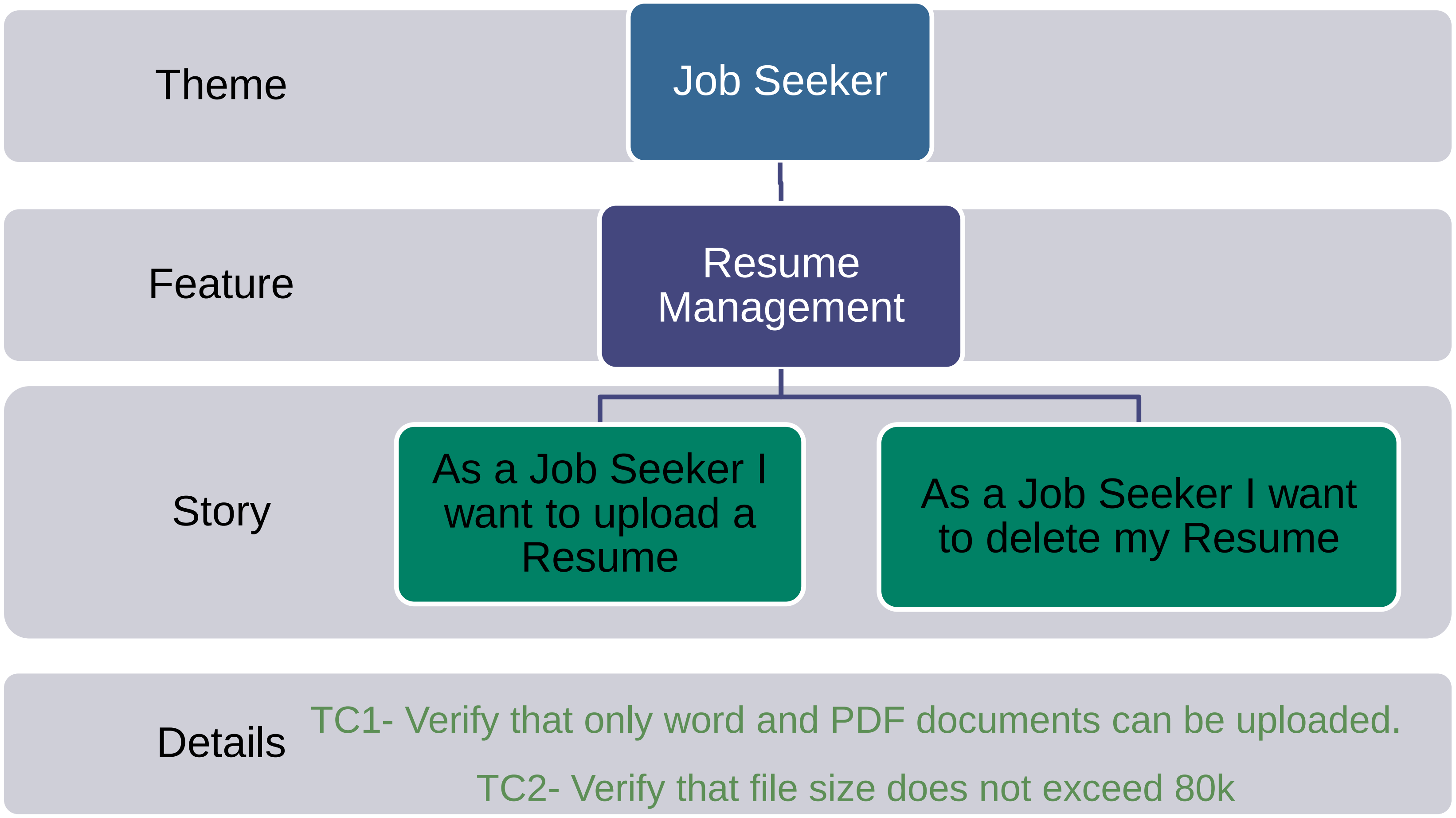
From Portfolio Ideas to Team Backlog

Levels of Agile Requirements



building talent, **driving** results

Example Requirement Levels


Theme
Job Seeker
Feature
Resume Management
Story
As a Job Seeker I want to upload a Resume
As a Job Seeker I want to delete my Resume
Details
TC1- Verify that only word and PDF documents can be uploaded.
TC2- Verify that file size does not exceed 80k

Please Take the Poll

- What level of Agile requirements development do you have the most challenges with?
  - Theme/Epic
  - Feature
  - User Story
  - Task

Requirements at the Portfolio Layer

Highest Level of Requirements

- In Agile, there are 2 “high level” requirements

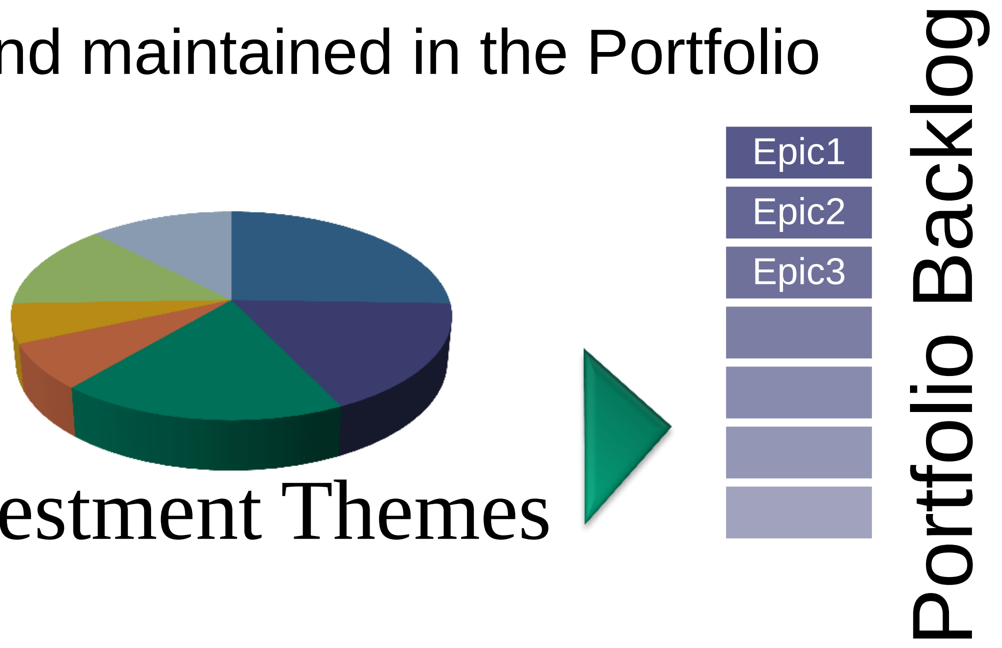
- Themes
- Epics

Epics – Portfolio Level

- Are large scale in nature and are used to realize investment themes
- Are the highest level of requirement
- Demonstrate **VISION** NOT specificity
- They must be categorized, prioritize and estimated
- They represent the “2nd layer of abstraction” in Themes

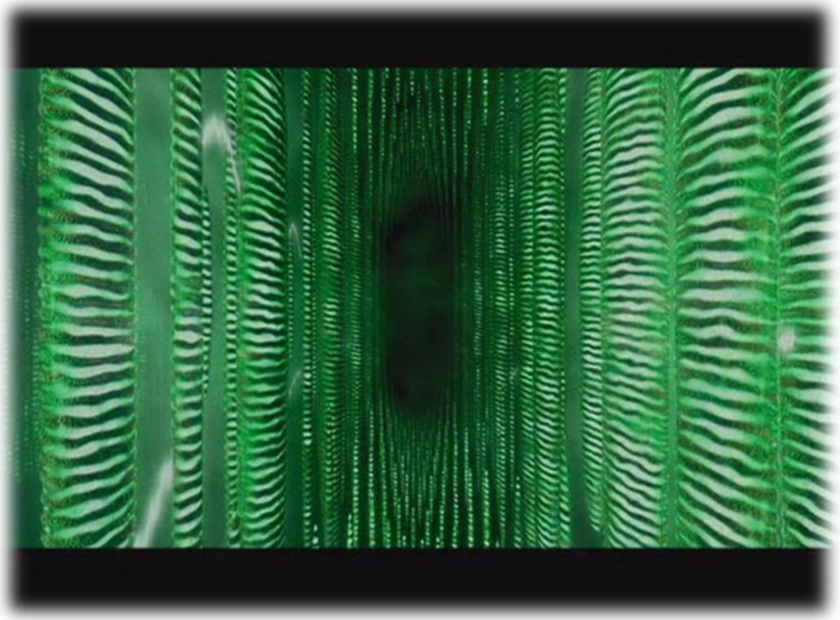
Epics – Portfolio Level

- Epics are managed and maintained by the Portfolio management group, governance team or business unit owners
- Epics are managed and maintained in the Portfolio backlog



Architectural Epics—Portfolio Level

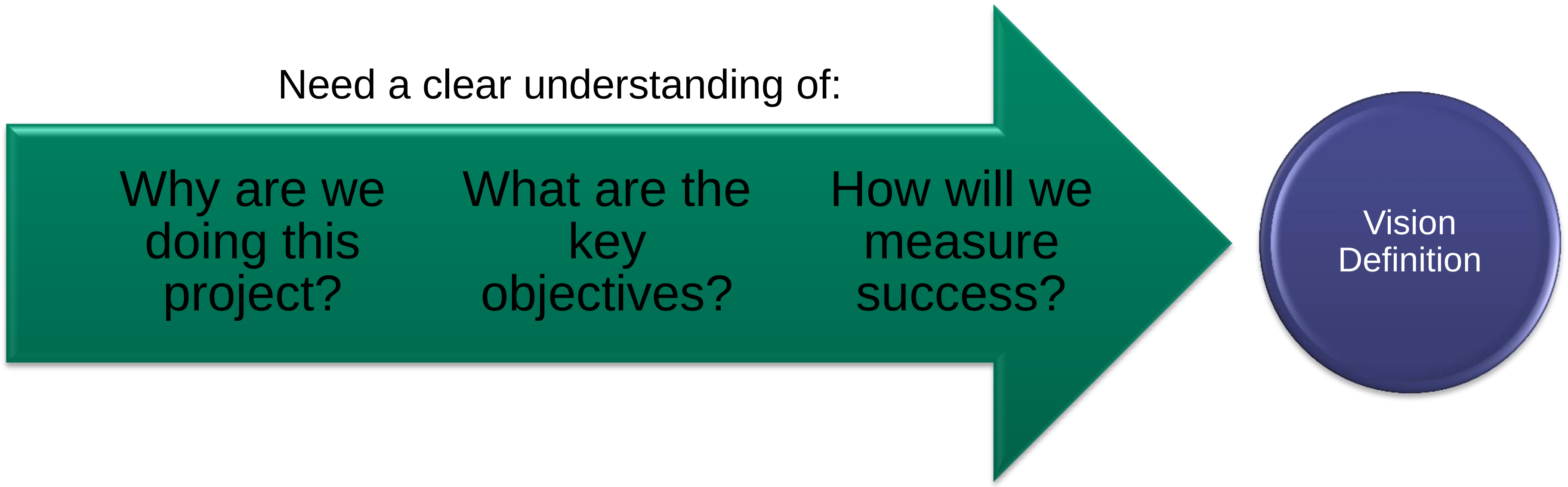
- Represent the technological/ infrastructure side of initiatives
- Architectural epics represent large scale implementations that could require execution in Waterfall practices as well as Agile



Understanding the Problem Domain

- Learn “Just Enough” about the problem domain.
- Identify actual users and impacted stakeholders.
- Go-See and understand **“What do they do today? Why is it not working?”**
- Use interviews, shadowing, surveys, and market research to understand true needs.

Visioning Summary


Need a clear understanding of:
Why are we doing this project?
What are the key objectives?
How will we measure success?
Vision Definition

Start collecting high level themes and features here that lead to defining the scope of the effort

Requirements Visioning

Goal

- Aims to identify a list of **major processes** and high level process steps for the system.
- **Who** needs to do **What**?
- Big picture of **UI Flow**

Techniques

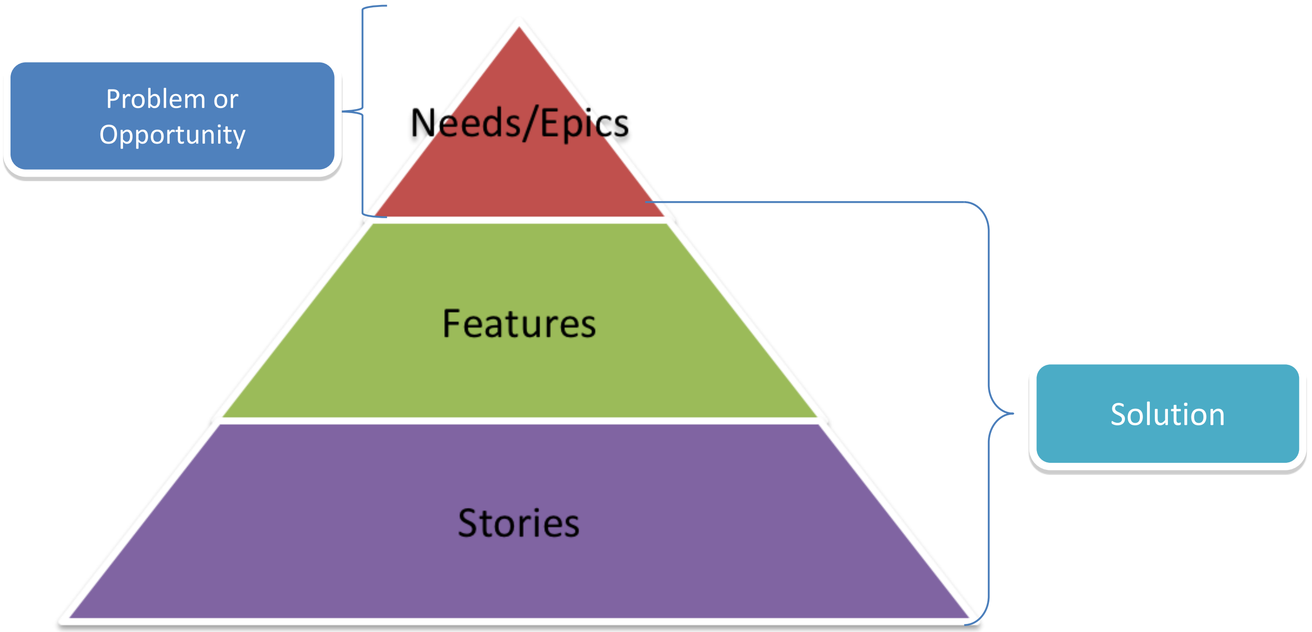
- Facilitator Led Visioning Workshop
- High Level Modeling
- Teams Breakout Converge

Output

- Use Case Diagrams
- High Level Process Diagrams
- UI Flow Diagrams
- Backlog (high level)

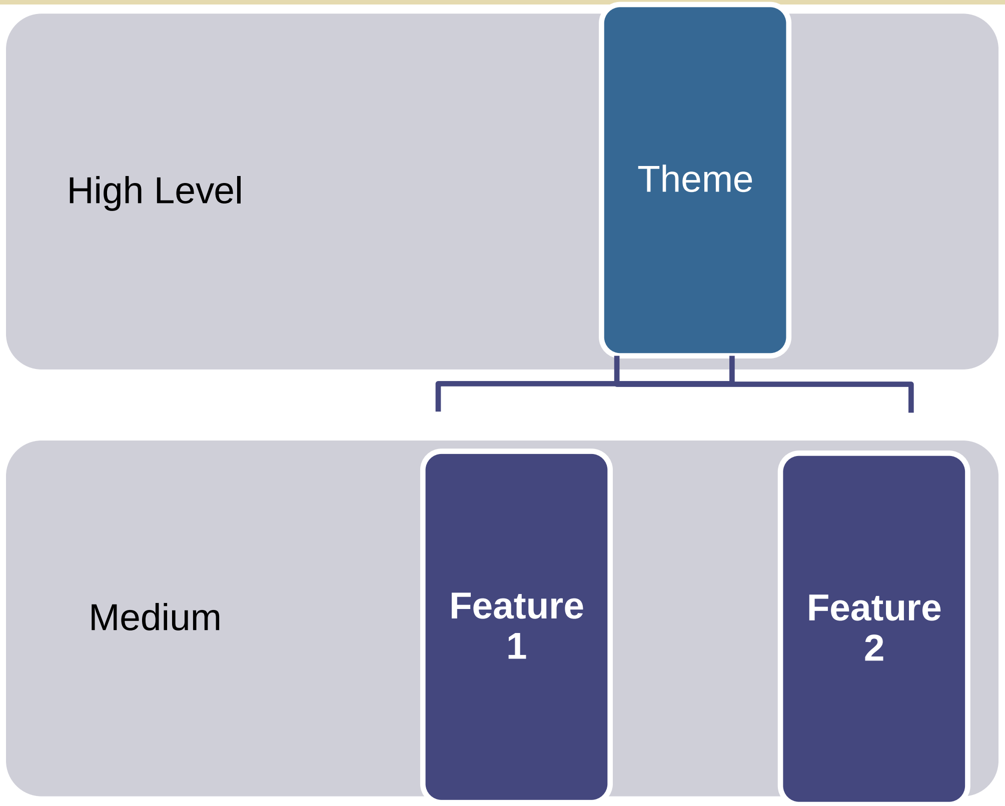
Features – Program Level

- What new things will a system do for its users
- Describes the benefits the users will get out of features
- They bridge the gap between "needs" and software requirements



Features – Program Level

- Features are a high level user story
- 25-50 features ensures a "broad view" of the product vision



Features are important for Agile teams to plan and estimate iterations

Challenges in Prioritizing Features for the Program Backlog

- Customers want them all
- Product managers want to avoid prioritizing features for sake of having them all
- Quantification of value for simple must haves is difficult to do for sake of remaining competitive – this is often too abstract in nature to prioritize
- Comparing "apple" features to "oranges" features can be very challenging

What is a Story?

- A brief, simple requirement statement from the perspective of the user
- Stories should be documented and visible
- Each story should have acceptance criteria

Attributes of a Story:

- **U**nderstandable
- **I**ndependent
- **N**egotiable
- **V**aluable
- **E**stimable
- **S**mall
- **T**estable

User Stories – Team Level

Characteristics of a great user story:

- *They are short and easy to read*
- *They are captured in a "list format" large BRD's not welcome here!*
- *They can be discarded after implementation*
- *They represent small increments of functionality*
- *They should be relatively easy to estimate*
- *Detailed system behavior is NOT captured in a user story*

The best litmus test of user stories is the use of the mnemonic UINVEST

User Stories – Team Level

Characteristics of a great user story;

- ***Card***
  - *2 or 3 sentences to describe intent of story*
  - *Format: As a <role> I can <activity> so that <business value>*
- ***Conversation***
  - *The card in essence is the introduction to a conversation between, product owner, developers, users, team, customer etc, in short all stakeholders involved. The conversation is intended to seek clarity* ***and drive out details***
- ***Confirmation***
  - *= Acceptance test, has the story been implemented according to conditions of satisfaction?*

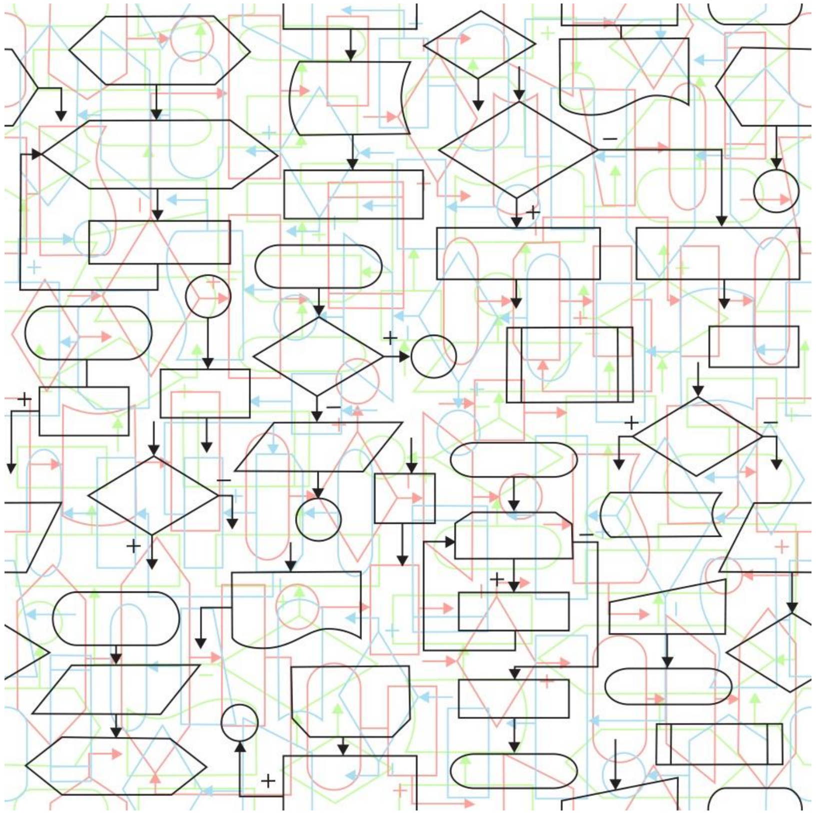
Requirements Writing Workshops

- Involve as many team members as possible.
- Goal is to brainstorm and write as many user stories as possible.
- Prepare the room with post-it notes, flip charts and markers.
- Need an effective facilitator to run these meetings to keep folks on track. **Establish Meeting Norms!**

Identifying Stories

Depending on your project type, there are several ways to identify stories such as:

- User Focused: Use Case Diagrams
- Process Focused: Process Diagrams
- UI Focused: UI Flow Diagrams



Please Take the Poll

- When developing Agile requirements, what modeling techniques do you use?
  - Use Case Modeling
  - Process Modeling
  - Business Information Modeling
  - None

Personas

- Personas are like 'avatars' that represent different customer segments for your business. They stand-in for 'real' users.
- We use them to learn about characteristics, needs and behaviors of real users.

High Level Use Case Diagrams

What Are They?

- Diagrams that demonstrate the Actors and their Goals. Actors can be people or systems.

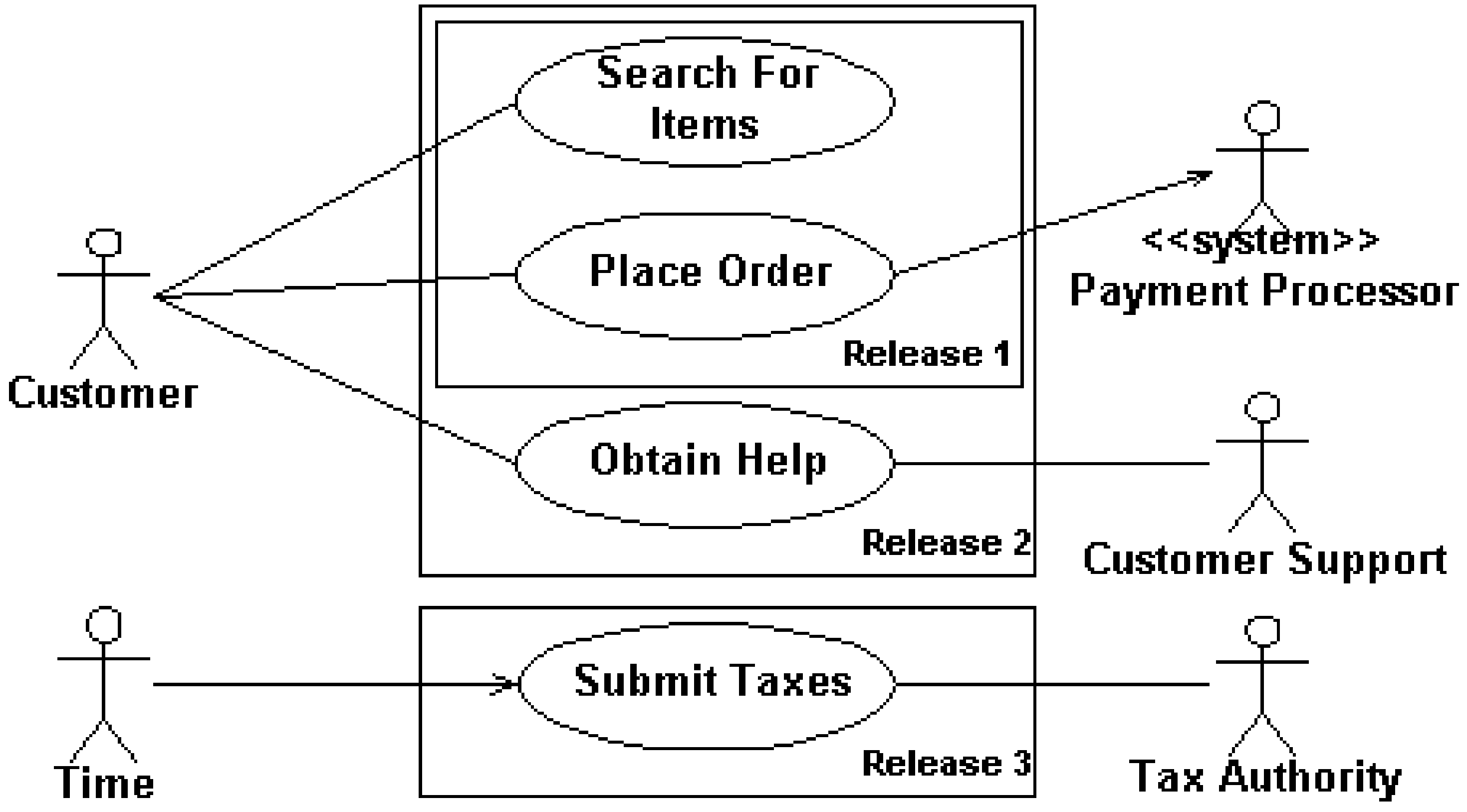
When to Use Them?

- For high level visioning.
- When identifying Themes and Features.
- For communicating a simple visual representation of the project scope.

Steps for Identifying User Roles

- Research the domain and interview real users.
- Perform surveys to identify true needs, behaviors and characteristics.
- Brainstorm the initial set of roles with product owner and SMEs.
- Organize the set.
- Consolidate and prioritize the target roles.

Sample Use Case Diagram

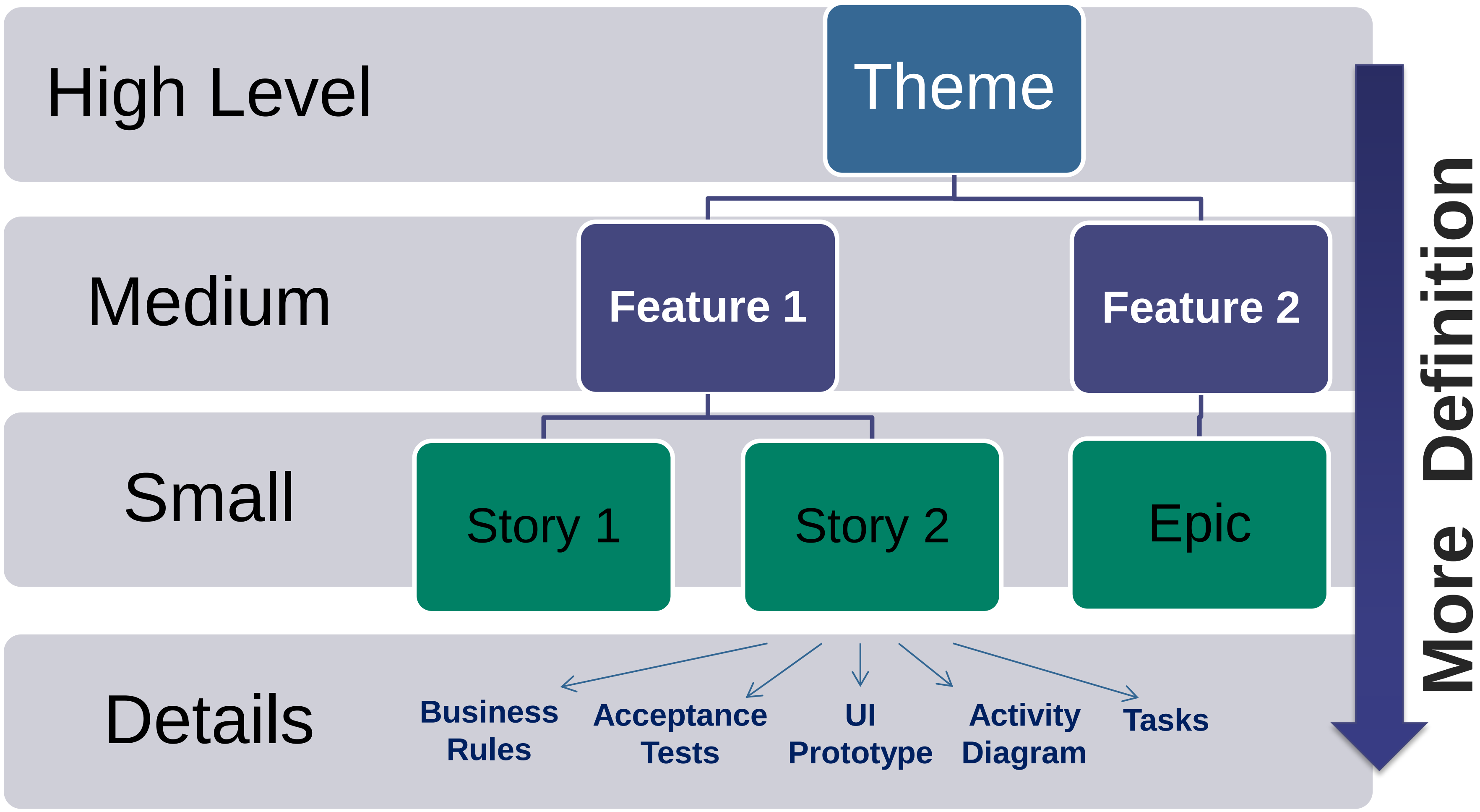


Scott Ambler
www.AgileModeling.com

Sample Questions to Ask

- What are you trying to achieve? Why?
- Who is involved, and how?
- What do those people want? Do they agree?
- How do you envision this working?
- What could go wrong?
- Why are you making these decisions?
- What are you assuming?

The Levels of Requirements



Example Theme–Feature–Story

Employer Area	Manage Jobs	1. As an employer I want to post a job so others can find it.
		2. As an employer I want to modify a job posting so it is correct.
		3. As an employer I want view a list of my open job postings so I can analyze them.

Example Story–Acceptance Criteria

1. As an employer I want to post a job so others can find it.	1. UAT1–Verify that only an authorized user with a valid employer account can post a job.
	2. UAT2–Verify that a duplicate job posting cannot be entered.
	3. UAT3–Verify that the posting date is past today's date.
	4. UAT4–Verify that the positing expiration date within 90 days.
	5. UAT5–Verify that the screen fields pass our standard field format rules (link here to doc).
	6. UAT6–Verify that all required fields are entered (list them or link to UI Prototype).

Example Story–Tasks

1. As an employer I want to post a job so others can find it.	**1.** Create a database table to store the job posting details.
	2. Design and build the screen for job posting.
	3. Write the automated acceptance tests
	4. Code, unit test and automate UAT1
	5. Document/record the on page video help for the job posting page.
	6. Perform user acceptance testing.
	7. Deploy the code to the test environment.
	8. ….. Others as needed.

Agile Requirements Elicitation Techniques

Visioning
- Interviews/Surveys
- User Roles, Personas
- Use Cases Diagrams
- Process Diagrams
- UI Flow Diagrams
- Context Diagrams

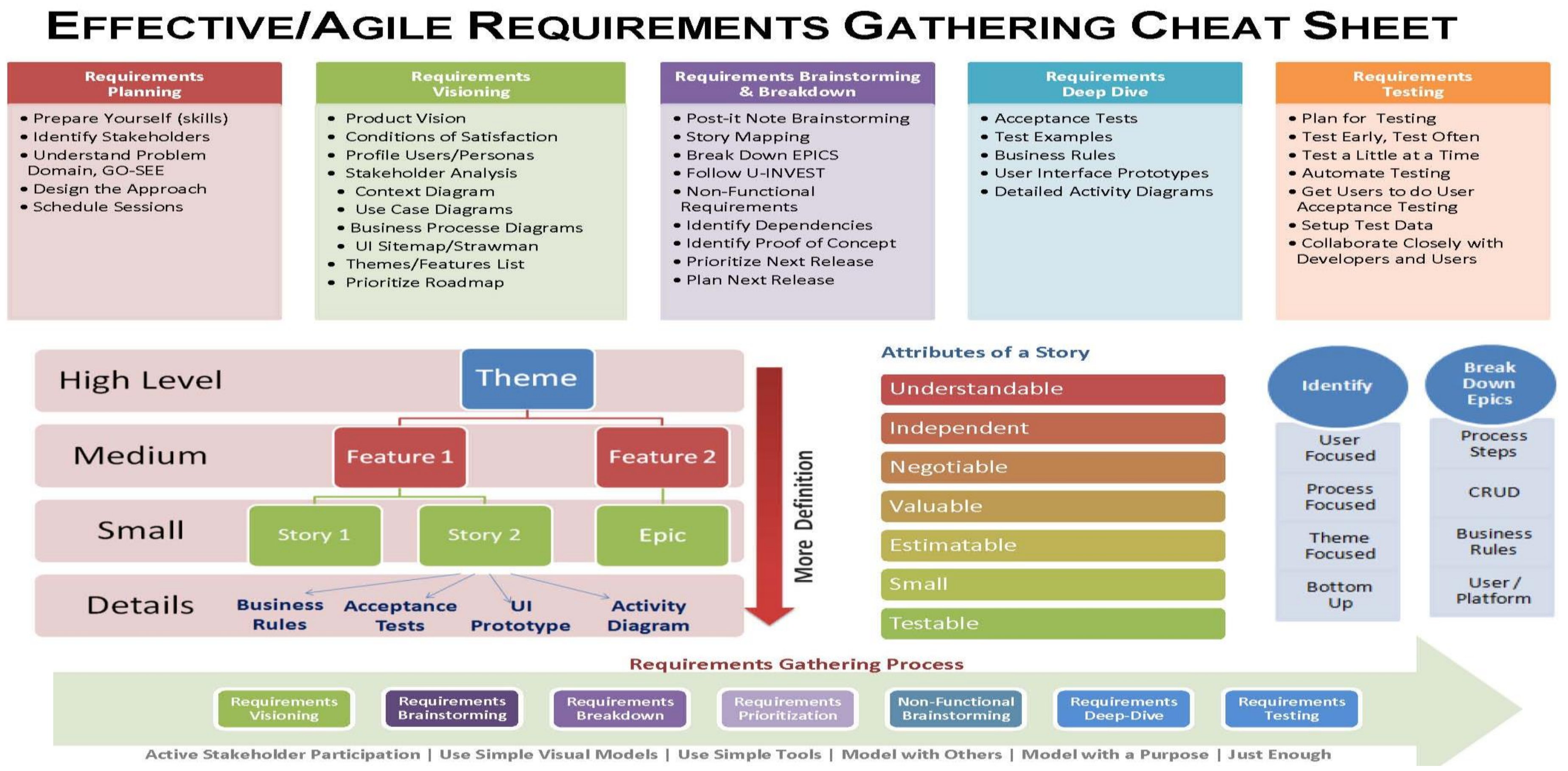
Brainstorming
- Group Brainstorming
- Facilitator Led Callout
- Post-it Note
- Breakout/Converge
- Story Mapping
- Silent Sorting

Breakdown/Slicing
- CRUD
- Business Rule
- Process Steps
- User/Platform

Deep Dive
- Acceptance Tests
- Test Scenarios
- Example Tables
- UI Prototyping and Wireframes
- Business Rules
- Activity Diagrams

Pulling It All Together

Levels of Agile Requirements



Effective/Agile Requirements Gathering Cheat Sheet

Requirements Planning
- Prepare Yourself (skills)
- Identify Stakeholders
- Understand Problem Domain, GO-SEE
- Design the Approach
- Schedule Sessions

Requirements Visioning
- Product Vision
- Conditions of Satisfaction
- Profile Users/Personas
- Stakeholder Analysis
  - Context Diagram
  - Use Case Diagrams
  - Business Processe Diagrams
  - UI Sitemap/Strawman
- Themes/Features List
- Prioritize Roadmap

Requirements Brainstorming & Breakdown
- Post-it Note Brainstorming
- Story Mapping
- Break Down EPICS
- Follow U-INVEST
- Non-Functional Requirements
- Identify Dependencies
- Identify Proof of Concept
- Prioritize Next Release
- Plan Next Release

Requirements Deep Dive
- Acceptance Tests
- Test Examples
- Business Rules
- User Interface Prototypes
- Detailed Activity Diagrams

Requirements Testing
- Plan for Testing
- Test Early, Test Often
- Test a Little at a Time
- Automate Testing
- Get Users to do User Acceptance Testing
- Setup Test Data
- Collaborate Closely with Developers and Users

Attributes of a Story
- Understandable
- Independent
- Negotiable
- Valuable
- Estimatable
- Small
- Testable

Identify	Break Down Epics
User Focused	Process Steps
Process Focused	CRUD
Theme Focused	Business Rules
Bottom Up	User / Platform

Requirements Gathering Process

Requirements Visioning → Requirements Brainstorming → Requirements Breakdown → Requirements Prioritization → Non-Functional Brainstorming → Requirements Deep-Dive → Requirements Testing

Active Stakeholder Participation | Use Simple Visual Models | Use Simple Tools | Model with Others | Model with a Purpose | Just Enough

building talent, **driving** results

Thanks for listening!

Peter Schmidt, PMP, PMI-ACP, CPL
VP, Client Services, ESI International
pschmidt@esi-intl.com

www.esi-intl.com