Epsilon Systems Presents:

An Overview of the Scrum Framework

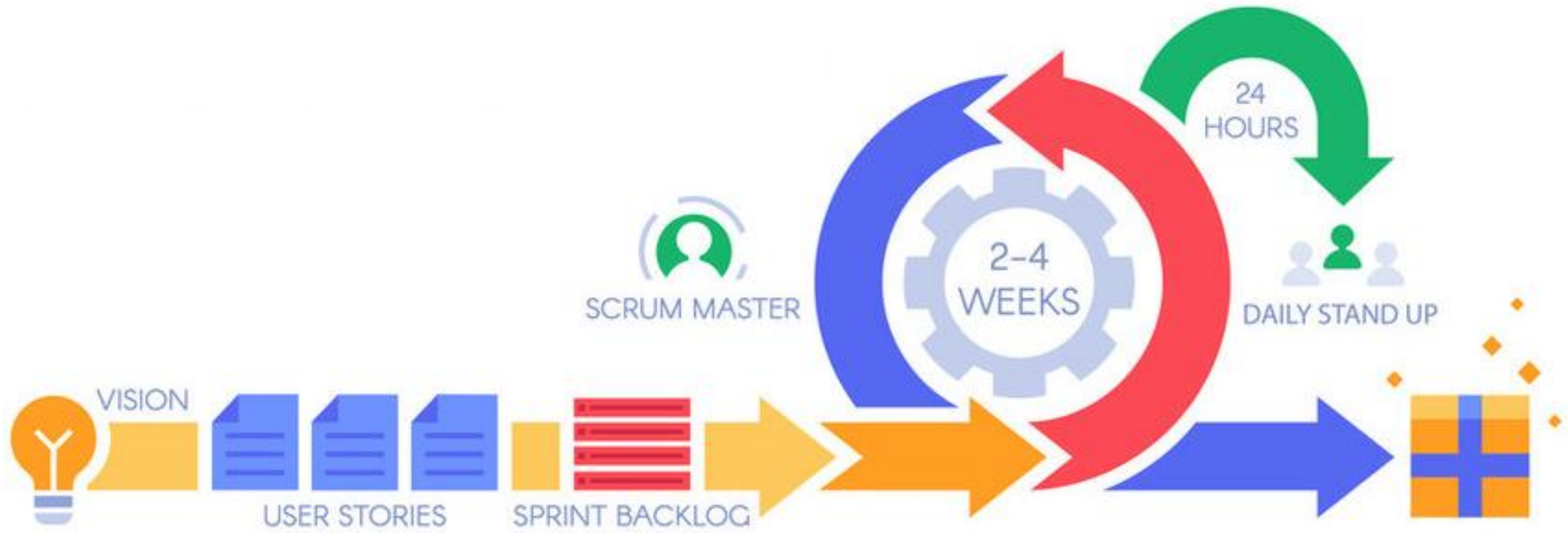
EPSILON® SYSTEMS

A 100% Employee-Owned Company

What is Scrum ?

- Scrum is a democratized **FRAMEWORK** … not a *Methodology*

- Scrum is an Agile framework that helps deliver value by completing challenging projects. Scrum uses cross-functional, self-organizing teams to work in short cycles to deliver products and services. The goal of Scrum is to help teams work together to delight customers.



Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

- **Individuals & Interactions** … over … *Processes & Tools*
- **Working Software** … over … *Comprehensive Documentation*
- **Customer Collaboration** … over … *Contract Negotiation*
- **Responding to Change** … over … *Following a Plan*

That is, while there is value in the items on the right, we value the items on the left more.

Scrum Values

SCRUM VALUES

COURAGE
Scrum Team members have courage to do the right thing and work on tough problems

FOCUS
Everyone focuses on the work of the Sprint and the goals of the Scrum Team

COMMITMENT
People personally commit to achieving the goals of the Scrum Team

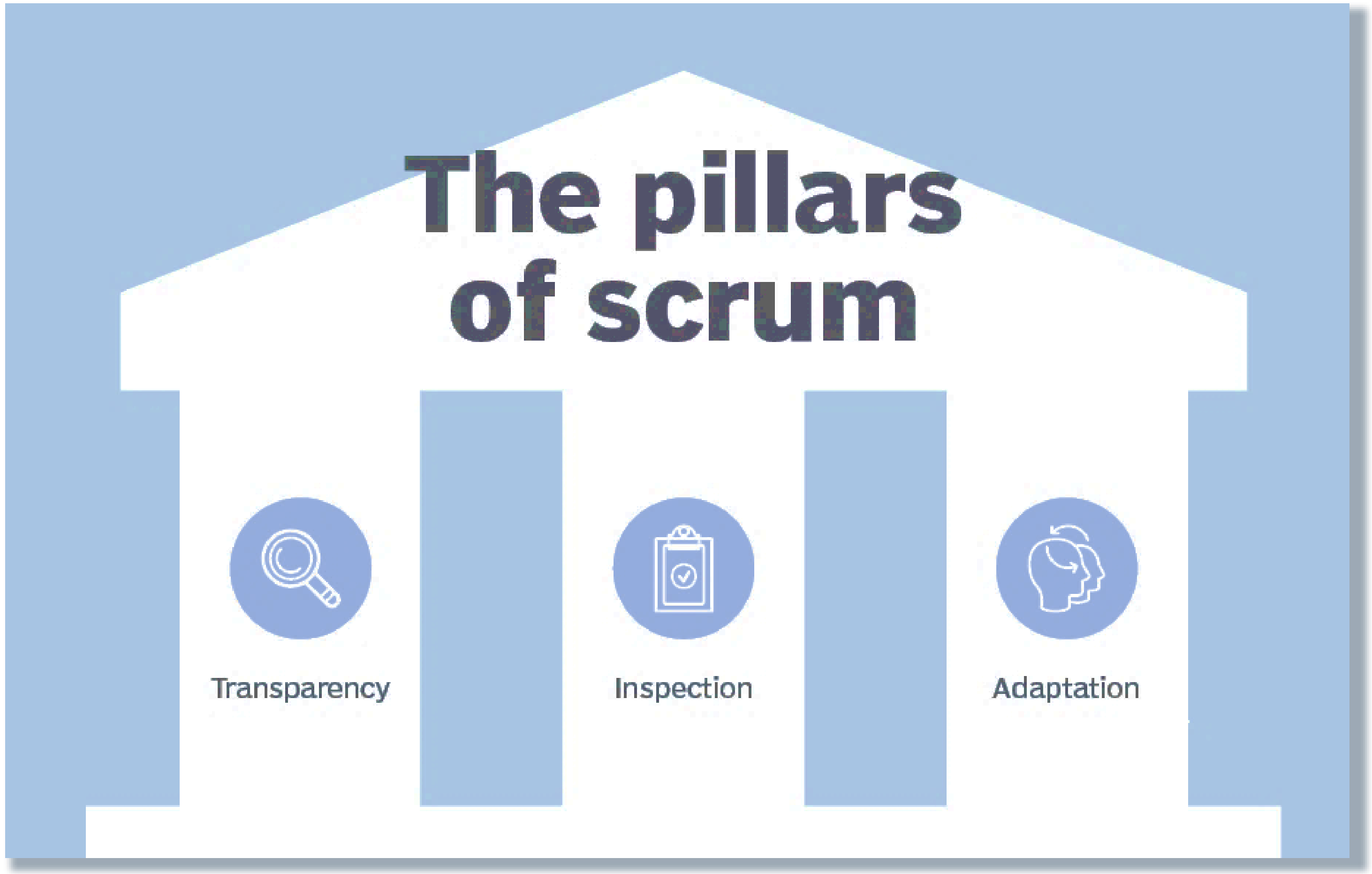
RESPECT
Scrum Team members respect each other to be capable, independent people

OPENNESS
The Scrum Team and its stakeholders agree to be open about all the work and the challenges with performing the work

Empirical Pillars of Scrum

- **Transparency**
- **Inspection**
- **Adaptation**

- Scrum IS … **Iterative**
- Scrum is ALSO … **Incremental**

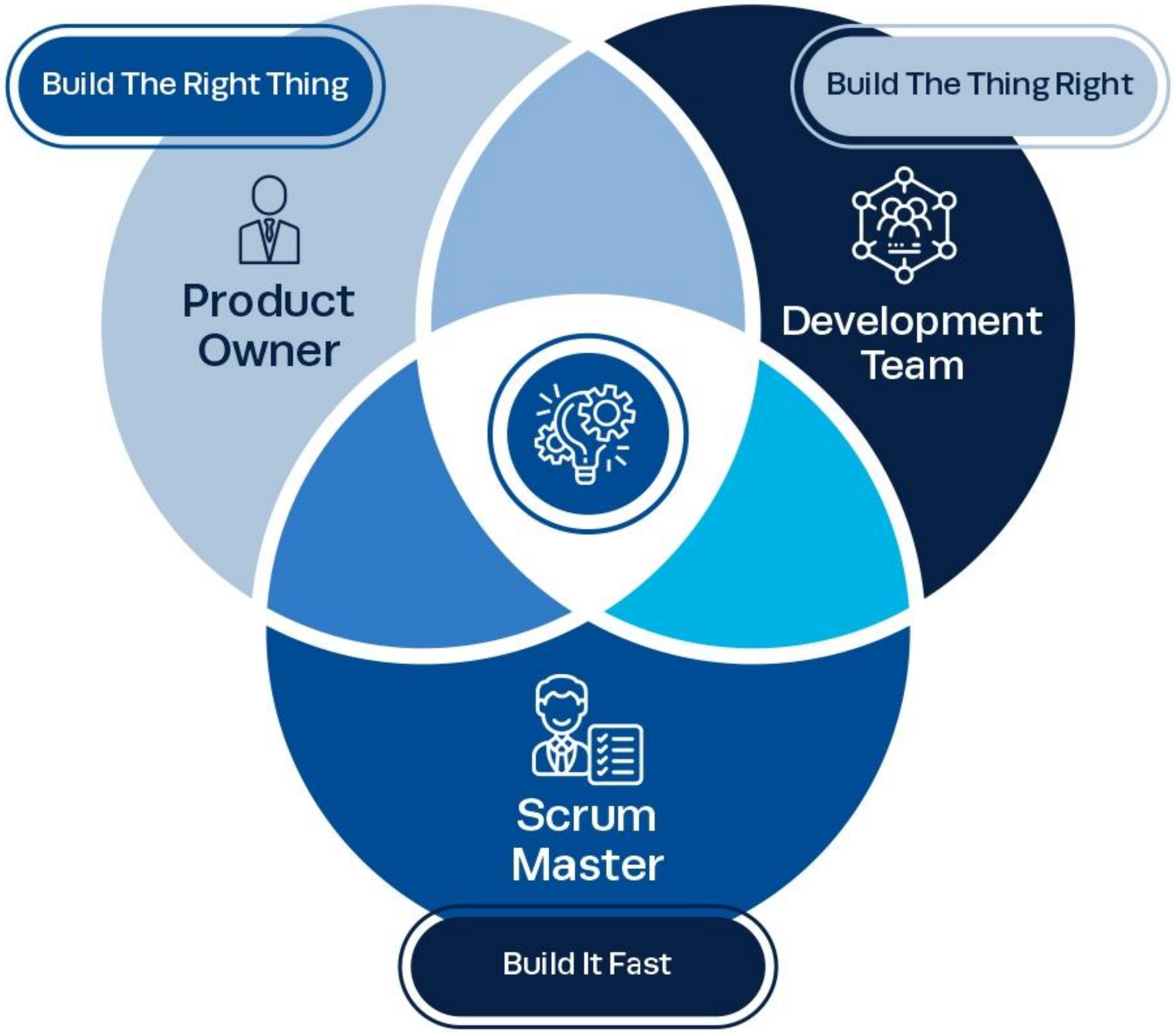


Scrum Events

- Sprint Planning
- Sprint
- Daily Standup
- Sprint Review
- Sprint Retrospective

Scrum Roles | Primary Focus

- **Product Owner**
  - Defines the Vision
- **Developers**
  - Executes the Vision
- **Scrum Master**
  - Empowers the Team



Product Owner | Role & Accountabilities

- Understands & Advocates for the needs & wants of the Stakeholders filtered through any & all organizational mandates.
- Communicates decisions & rationale clearly to the Developers & Stakeholders.
- Understand & Communicates the overall Vision in a compelling way to the Team, always connecting it to the development process.
- Demonstrates their commitment to the team by maintaining a robust & dynamic Product Backlog, spending the most time adding, reviewing & prioritizing items.
- The Product Backlog empowers Developers to Execute the Vision
  - Always having Value-Added work to do
  - Remaining Focused throughout each Sprint

WHO ?

WHAT ?

Developers | Role & Accountabilities

- **Roles**
  - Autonomous
  - Accountable
  - Self-Directed
  - Cross Functional
  - Goal-Driven
  - Delivery-Obsessed
- **Accountabilities**
  - Thorough Understanding the Requirements
  - Providing Inputs for & Estimates of User Stories
  - Communicative. Immediately Escalating any Issues
  - Instilling quality by adhering to a Definition of Done
  - Raise awareness of all potential technical dependencies

When ?

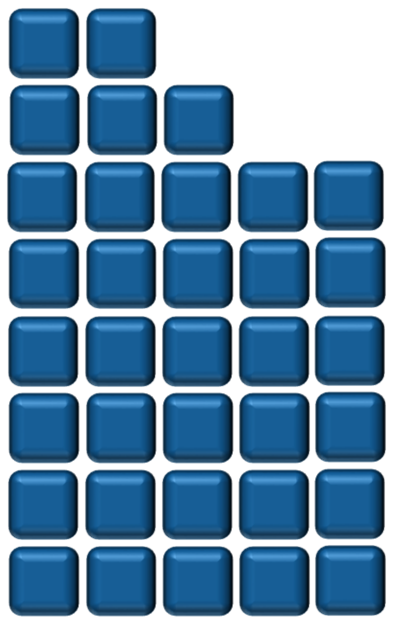
How ?

Scrum Master | Role & Accountabilities

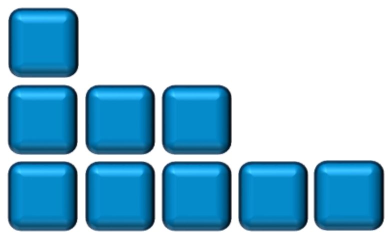
- **Facilitate the Scrum Process**: Ensures Scrum is rigorously followed in everything the team does.
- **Remove Impediments**: Responsible for identifying & removing anything preventing the team from delivering value.
- **Coach the Team**: Responsible for coaching the team on Scrum principles & practices. This includes helping the team continuously improve their processes.
- **Protect the Team**: Responsible for protecting the team from outside distractions, allowing the team to focus on their work and deliver value.
- **Facilitate Communication**: Responsible for ensuring that there is clear communication between the team, the Product Owner, and any other stakeholders.
- **Ensure Transparency**: Responsible for ensuring that the team's progress is transparent to all stakeholders. This includes maintaining a visible product backlog and burn-down chart.

Artifacts & Commitments

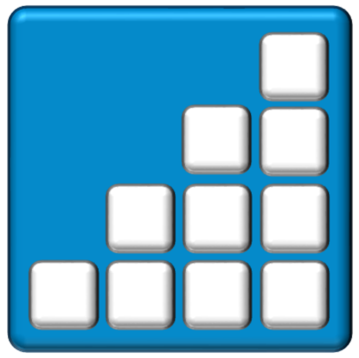
- **Product Goal**
  - Product Backlog
    - Product Backlog Items
- **Sprint Goal**
  - Sprint Backlog
    - Epics
    - User Stories
    - Tasks
- **Increment**
  - Definition of Done



Product Backlog



Sprint Backlog

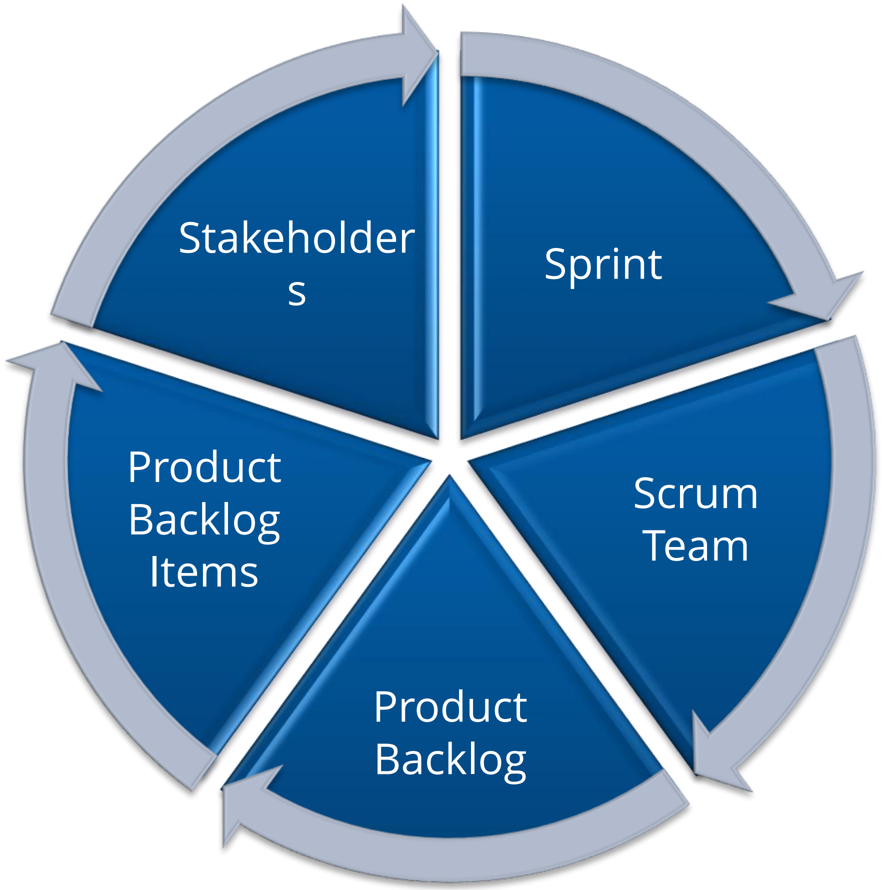


Increment

NASIC DoD: *All PED Mod product teams shall conform to the extended overarching PED Mod Definition of Done for product backlog items, incorporating known best practices, as defined: Containerized microservices follow the 12-Factor Methodology for cloud-native design and implementation (https://12factor.net/). Unit test coverage of >80%. All code peer-reviewed. All acceptance criteria met. Technical design and interface information updated. User guide instructions updated for features and capabilities. Integration tests successfully executed for all implemented features. Code and dependencies properly versioned and managed. Documentation for developers and maintainers is complete and up-to-date*

Product Backlog Refinement | Defined

- The **SPRINT** is a value-creating MACHINE:
  - *Raw crude oil goes in, and refined fuel comes out.*
- The **SCRUM TEAM** is the ENGINE:
  - *The engine makes the Scrum machine run.*
- The **PRODUCT BACKLOG** is the GAS TANK:
  - *An engine needs fuel to run, and something to hold that fuel.*
- The **PRODUCT BACKLOG ITEMS** are the GAS in the tank:
  - *No gas, no value.*
- The **STAKEHOLDERS** are the GAS STATION:
  - *No User Stories, no gas*



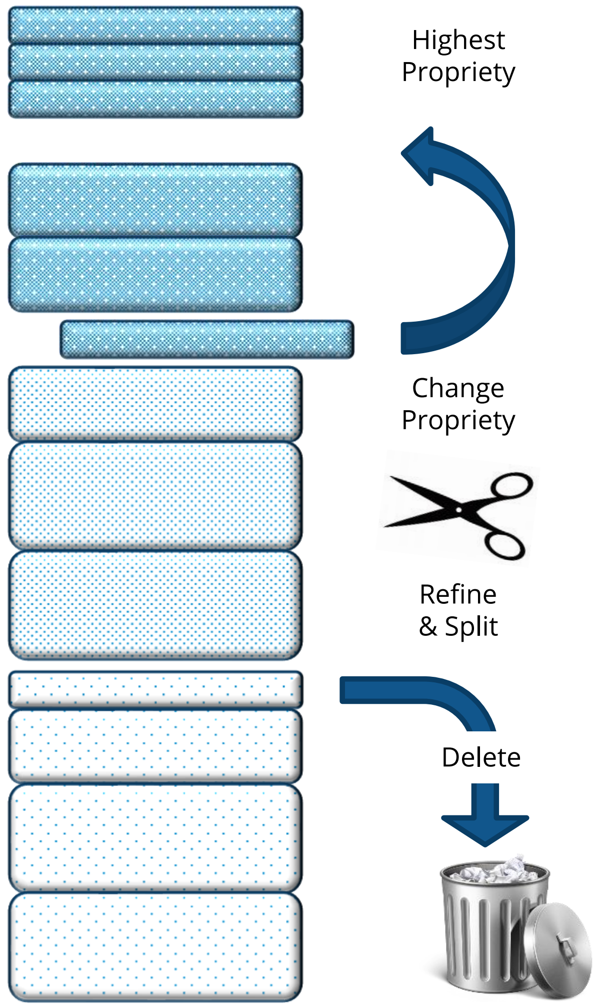
Product Backlog Refinement | Roles

- **Product Owner**
  - Continuous refines activity taking a full 30% of their time
  - Advocates for the needs of Stakeholders & Organization as a whole
  - Changes & Updates priority, details of work, description, and priority
  - Refines PBI's to get them "ready" for upcoming Sprint(s).

- **Developers**
  - Developers collaborate with and support the Product Owner in refinement.
  - Provide estimates of the effort required for each PBI & helping refine the Acceptance Criteria.
  - Time spent should be limited to keep Developers focused on their work in the current Sprint.
  - Not everyone needs to participate.

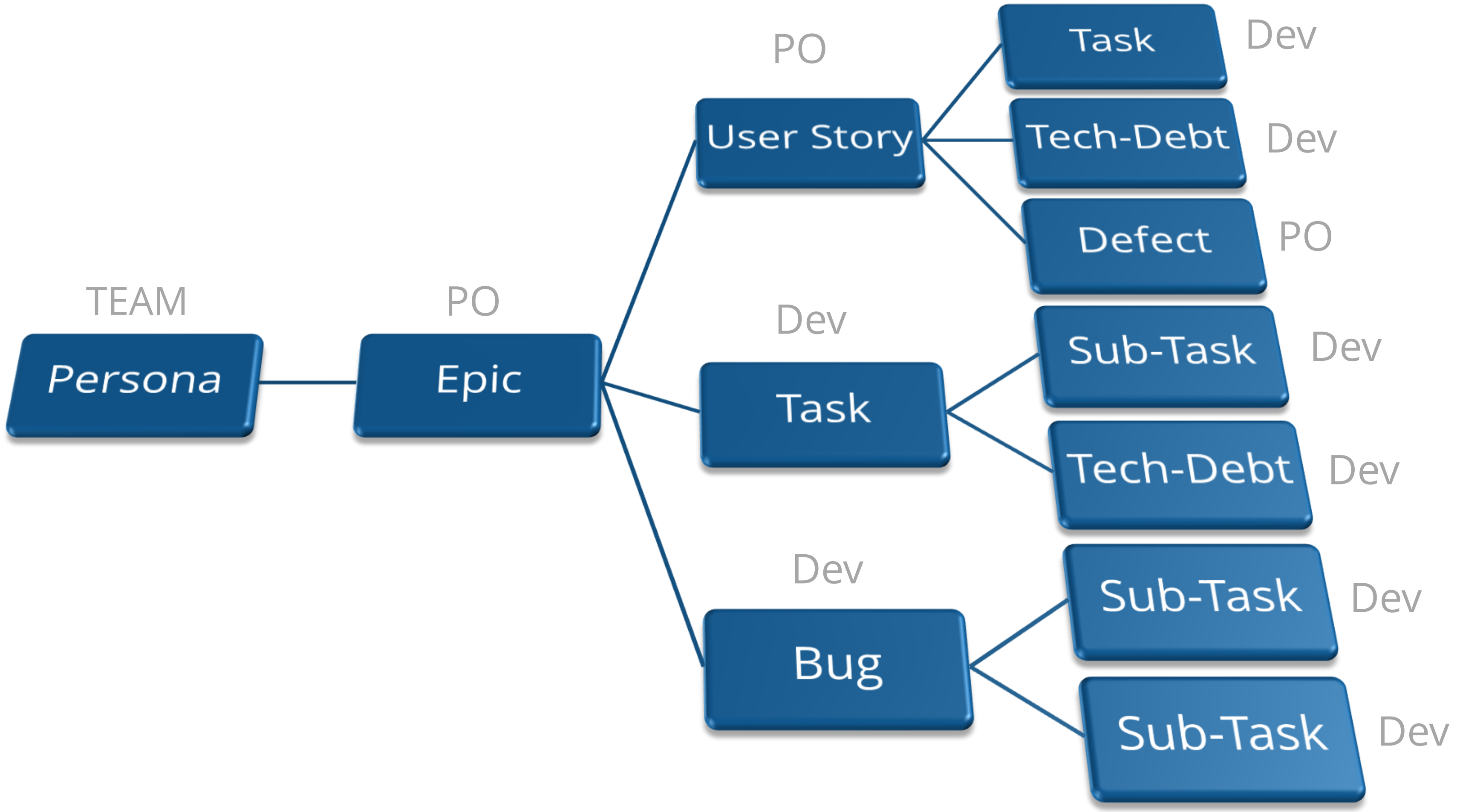
- **PBI's** are expressed with these attributes
  - Story Format & Description
  - Value & Priority
  - Estimate

- Does the entire Team have a "*shared definition*" of "**READY**" ?



Product Backlog Hierarchy

- Persona
  - Epic
    - User Story
      - Task
      - Tech-Debt
      - Defect
    - Task
      - Sub-Task
      - Tech-Debt
    - Bug
      - Sub-Task
      - Sub-Task



Tech Debt | Bug Fixes | Rework

- **Technical Debt** is the accumulated consequences of developer decisions; shortcuts, trade-offs, compromises in the design & implementation of software systems.
  - <u>Examples</u>: refactoring code, improving design patterns, or dedicating specific development cycles to address underlying issues.
  - Can negatively impact the entire project

- **Bug Fixes** are specific instances of code errors that need to be directly addressed to ensure proper functionality
  - Typically no significant impact to the system & addressed as they are identified
  - <u>Example</u>: Correcting a typo in a variable name that is causing a calculation to produce incorrect results.

- **Rework** refers to the process of modifying or redoing existing code, design, or other project elements after work has already begun, usually due to discovered errors, changing requirements, or poor initial design, resulting in additional development effort beyond the initial implementation.
  - The effort required to fix, modify, or improve software systems due to defects, changes, or feedback.

What is a User Story ?

- **As a** *<type of user>*,
- **I want/need to** *<goal or action>*
- **So that I** *<value created>*

- **Gherkin Syntax**
  - A set of keywords to give structure &meaning to executable specifications
    - Given
    - When
    - Then

- **4 C's**
  - A concise outline to remove the jargon or fluff from user stories, in order to produce something clear enough to be understood by anyone.
    - Conversation – *just enough detail to create shared understanding*
    - Card – *describe a user, task and goal in 3 statements*
    - Confirmation – *rules for agreeing that a User story is done; Acceptance Criteria*
    - Context – *deeper understanding and insight on how it related to other User Stories*

I.N.V.E.S.T.ment Makes a Good User Story

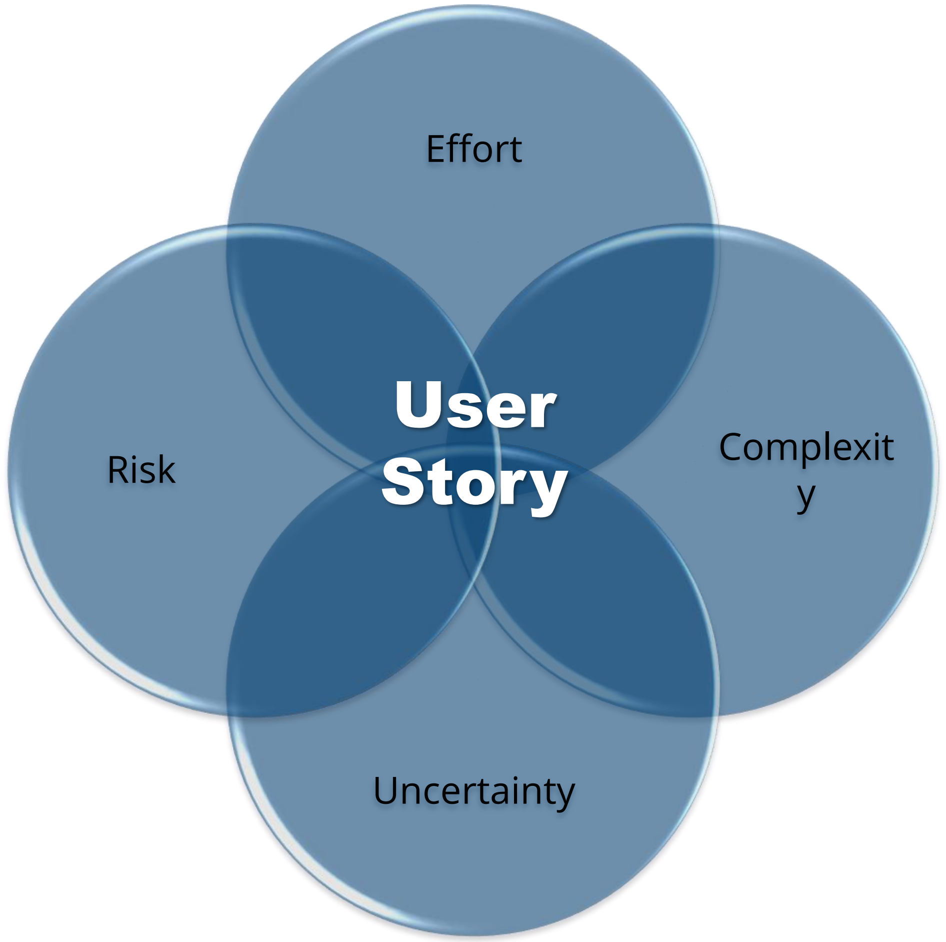
- **Independent**
  - *They must not be allowed to overlap in concept.*
- **Negotiable**
  - *Users & Developers co-create. Its about essence, <u>not</u> details. Not an explicit contract for features.*
- **Valuable**
  - *Value as the User would define it, taking into account every layer & its dependencies.*
- **Estimable**
  - *Doesn’t have to be exact. You can’t estimate without mutual understanding and that’s negotiated.*
- **Small**
  - *No more than a few person-weeks worth of work.*
- **Testable**
  - *Make sure you & the customer understand it enough to test for it.*

S.M.A.R.T. Tasks Focus Effort

- **Specific**
  - *User Stories are easier to understand & work if they don’t overlap in concept.*
- **Measurable**
  - *Everyone needs to agree on when can we mark it done ?*
- **Achievable**
  - *Ensure everyone is up to the task*
- **Relevant**
  - *All work must contribute to the User Story at hand.*
- **Time-Boxed**
  - *Limits needless rabbit-trailing …*

What is a Story Point ?

- A story point is a unit of measurement for a User Story and a quick way to estimate work.
- The intent is to help the team gain a shared understanding of the work relative to all user stories in the product backlog.
- Some teams establish an agreed-upon baseline for the lowest effort story with the lowest story point.
- If a User Story has a high number of points, the team breaks it into smaller stories and estimates each one.



What is Definition of Done ?

- The Definition of Done (DoD) **DEFINES** what must be true for the work to be considered “done”.
- It represents a **SHARED UNDERSTANDING** among all members of the team of the quality and completion criteria for the product.
- It’s a formal list of **CRITERIA** that describe the **QUALITY** measures required for the product.

- **NASIC DoD**

Acceptance Criteria vs. Definition of Done

- Specific to **ONE PBI** or **User Story**, described in conditions to validate a specific work item.
- Each PBI may have a different set of Acceptance Criteria.

- Applies to **ALL THE WORK** the team is doing. It describes the conditions to validate that the product Increment is usable.
- Must be satisfied by every PBI completed by the Scrum Team in the Sprint.