Deloitte Agile Framework

This framework highlights the **core elements** of an agile project but is not all inclusive.
Additional activities[2] may be required based on the context, scope and scale of the project.

Iterative & incremental cycle that is repeated to deliver value frequently

Project Planning
Typically, 1-3 weeks[1]

Initiate and plan the project

- Establish **ways of working** (Project Management Plan)
- Develop **schedules**
  - Master Plan
  - Integrated Work Plan
- Onboard and train team
- Define and configure **project tools**
- Determine **status reporting approach**

Discovery
Typically, 1-2 months[1]

Initiate product backlog and establish foundational architecture & infrastructure

- Establish user experience approach and personas
- Define and prioritize **EPICS** and **FEATURES**
- Establish solution architecture
- Develop initial **ROADMAP** and **MVP RELEASE PLAN**
- Define initial **DOR** and **DOD**
- Develop prioritized **PRODUCT BACKLOG**
- Decompose priority features to initial (and ready) **USER STORIES**
- Establish planned **CAPACITY** and **VELOCITY**
- Establish **Testing** (incl. automation) **strategy**
- Establish **Continuous Integration process**

Sprint Cycle(s)
Typically, 2-4 week sprints (shorter is better)

Iteratively develop the product through minimum viable product sets (MVP) in potentially deployable increments, and increase efficiency and quality through frequent feedback and improvement loops

On-going Refinement
- Perform **PRODUCT BACKLOG REFINEMENT**
- Refine roadmap & MVP release plan

Conduct SPRINT PLANNING
- Validate expected velocity and establish capacity
- Identify prioritized, ready user stories
- Decompose user stories into tasks with estimated hours
- Commit to completing a set of user stories

PRODUCT INCREMENT

Close sprint
- Conduct **SPRINT REVIEW**
- Calculate actual velocity
- Conduct **SPRINT RETROSPECTIVE** & identify improvements to implement

Execute Sprint: Deliver Working Software
- Develop application components
- Perform code review
- Conduct unit tests
- Produce software build
- Produce essential documentation
- Validate/test user story
- Obtain PO acceptance of user story

Collaborate & Manage
- Manage **SPRINT BACKLOG**
- Manage **BURNDOWN CHARTS**
- Conduct **DAILY STAND-UPS**
- Conduct **SCRUM OF SCRUMS**
- Update roadmap and **KANBAN BOARD**
- Resolve **IMPEDIMENTS**

Prepare for and perform testing
- Execute integration & regression testing
- Prepare for additional release testing as applicable

Complete non-sprinted project activities[2]

Release
Frequent product increments

Deploy the solution and provide subsequent support

- Conduct **release testing** including regression, integration and user-acceptance testing
- Execute **dress rehearsal**
- Make **go/no-go decision**
- Execute **deployment plan**
- Deliver **change management and training**
- Facilitate **production support** activities
- Execute **solution hand-over**
- Close release

Embody the agile mindset

The application of DevOps principles, processes & techniques with agile delivery allows working software to be released more frequently

[1] *Varies depending on project complexities*

[2] *May include infrastructure, change management, security & controls, data, deployment planning etc.*

Agile Framework: Execution Detail

Notes: This is a representative view, not to timeline scale and not inclusive of all activities/deliverables (see EVD for the full method content and details)

Discovery project planning = one-time activities; Discovery sprint 0 is meant to be iterative as needed

Phases should overlap as appropriate – e.g. Non-sprinted activities may overlap with Discovery Sprint 0/ come before sprinting can begin

Discovery

Plan & mobilize engagement, initiate product backlog, and establish foundational architecture and infrastructure

Project Planning

Establish ways of working
- ► Develop project management plan and user story management plan
- ► Prepare discovery sessions
- ► Prepare for software development, migration, source code, and integration by developing appropriate strategies, processes, standards (including continuous integration process and unit test approach)

Establish security and controls approach
- ► Define security architecture
- ► Develop security, privacy, and controls approach
- ► Design & develop project team application security roles

Plan and mobilize project
- ► Develop project charter
- ► Complete deliverables log
- ► Develop master plan
- ► Develop integrated work plan
- ► Plan and conduct kick-off
- ► Acquire project team
- ► Install and configure tools
- ► Train project staff

Sprint 0

Define business needs
- ► Perform high-level process design

Plan solution validation*
- ► Stand up baseline solution
- ► Plan and prepare baseline solution

Develop prioritized BACKLOG
- ► Define and prioritize **EPICS** and **FEATURES**
- ► Decompose priority features into initial **USER STORIES**

Define DOR and DOD

Define deployment strategy

Develop initial ROADMAP and MVP release plan

Establish CAPACITY and VELOCITY

Establish application and integration architecture

Establish user experience design
- ► Define user experience guidelines
- ► Define user personas

Design data management, cleansing, architecture, governance & plan for data conversion

Establish security and controls design
- ► Develop identity & access management solution design
- ► Develop privacy roadmap, requirements and control activity framework
- ► Develop general IT, interface and data conversion control framework
- ► Define segregation of duties and sensitive access

Establish foundational technical infrastructure
- ► Define logical infrastructure
- ► Define technical architecture

- ► Define test strategy

Plan test automation
- ► Define test automation approach
- ► Define & develop test automation framework

Mobilize and engage communities
- ► Develop organizational change management strategy
- ► Design change campaign
- ► Develop business advocate network (BAN) approach

*Optional for projects using transfer solution or pre-configured solution

Sprint cycles

Iteratively and incrementally develop potentially deployable product increments, increasing efficiency and quality through frequent feedback and improvement loops

Conduct SPRINT PLANNING

Define and refine PROGRAM INCREMENT

Deliver user stories

Deliver working software
- ► Configure system
- ► Develop code & review
- ► Produce software build
- ► Conduct unit test
- ► Develop essential documentation & detailed business processes as required

Deliver security and controls
- ► Build and test identity and access management solution
- ► Implement privacy roadmap
- ► Build and test role-based security and security vulnerability
- ► Confirm implementation of business process, general IT, interface and data conversion controls

Perform functional & regression testing
- ► Develop test cases
- ► Identify test data & populate environment
- ► Develop and test automation scripts
- ► Conduct test

User stories accepted by PO as delivered

Accelerate future skillsets
- ► Develop learning curriculum
- ► Develop learning courseware and simulations

Personalize the change campaign
- ► Execute the change campaign

Activate business networks
- ► Engage business advocate networks
- ► Develop and launch super user program
- ► Transfer knowledge to solution owners

Refine backlog (PBR) and roadmap
- ► Manage product & **SPRINT BACKLOG**
- ► *(Continue to)* Define/Refine user stories
- ► Refine roadmap

Conduct agile ceremonies/events
- ► Conduct **DAILY STAND-UP**
- ► Perform **SCRUM OF SCRUMS**

Close sprint
- ► Perform **SPRINT REVIEW**
- ► Calculate actual velocity
- ► Perform **SPRINT RETROSPECTIVE**

Non-sprinted activities

Deliver data conversion
- ► Develop specifications
- ► Develop code
- ► Conduct technical unit and functional object tests

Prepare for release testing (system integration/ integration, mock conversion, parallel, regression, performance and stress, infrastructure)
- ► Develop test approaches, cases & plan
- ► Identify test data

- ► Manage integrated work plan

- ► Track value realization

Prepare for production deployment
- ► Define release go/no-go criteria
- ► Develop deployment plan
- ► Define batch jobs
- ► Develop batch job schedule

Plan hyper care
- ► Develop to-be service delivery approach
- ► Develop service delivery transition plan

Plan service delivery
- ► Perform service delivery framework assessment
- ► Develop to-be service delivery approach
- ► Document service delivery requirements
- ► Develop service delivery recommendations & resource estimate

Release

Test, Deploy, Stabilize, Operate & Optimize/ Innovate

Deploy minimum viable product (MVP) sets, provide subsequent support, inspect and adapt

Test

Execute release testing (system integration/ integration, mock conversion, parallel, regression, performance and stress, infrastructure, user-acceptance)
- ► Conduct test

Deploy

Accelerate future skillsets
- ► Deploy & evaluate learning

Deploy production solution
- ► Deliver data integration and conversion solution
- ► Produce software build
- ► Execute dress rehearsal
- ► Conduct production deployment go/no-go
- ► Execute production deployment
- ► Perform legacy system decommissioning

Stabilize & Operate

Perform hypercare
- ► Perform service desk function
- ► Conduct service delivery transition

Transition service delivery
- ► Implement continuous service improvement program
- ► Conduct service delivery training
- ► Transition solution documentation to support organization

- ► Maintain and operate infrastructure

Close project

Key tips

- Allocate appropriate time for core planning activities and discovery at the beginning of the project
- Involve an agile coach from the start
- Establish appropriate governance and controls - and balance that with the agile/ lean mindset
- Invest adequate time and effort in aligning across Deloitte and client project team on roles, responsibilities, delivery approach, scope, ceremonies, etc.
- Train agile team on agile mindset and discovery process
- Define your epic/feature structure early so there is consistency and so project reporting needs are met
- Understand forecasted scope vs capacity and timeline at high-level from the start to understand how much capacity you can dedicate to adjusting, iterating and making changes to the solution based on feedback
- Complete foundational elements such as establishing user experience design and technical infrastructure early to inform user stories
- Do just enough foundational solution design to support writing user stories with accuracy
- Invest energy developing solid user stories (and training the team to do so)
- Have enough 'ready' user stories (and foundational design) to get started (at least 3 sprints' worth), but not so much that you box yourself in/ struggle to flexible or require significant re-work
- Define a process for developing "ready" user stories
  - Product owner collaborates with Deloitte and client development team members to develop user stories, including required supporting documentation to make the "what" clear
  - Development team reviews, confirms understanding and estimates user stories
  - Product owner approves user stories
- In the roadmap, don't put off all the big or complex user stories until the end
- When prioritizing the product backlog, manage against the contracted scope (as relevant), have open conversations about tradeoffs, and manage delivery dates
- MVP requires a "critical mass" of components that have to be deployed in an initial release to provide a usable solution; additional advanced features or capabilities can be added later to enhance the product

Key tips

- The engagement work determined to be delivered in a traditional way, needs to be planned for, capacity allocated and managed in an integrated work plan or within the agile management tool as pseudo user stories
- Only sprint user stories that are ready (the team, using the DoR, decides which user stories are ready)
- Consider capacity when planning (ideal hours) the sprint backlog and managing the roadmap; dedicate enough capacity (outside of the sprint) for PBR and any other non-sprinted activities (e.g. test prep)
- Let development teams work independently as they sprint – no changes mid-sprint
- Design related documentation (e.g. specs) should be lean and focus on what's essential/ required
- Adopt Test-Driven or Behavior-Driven development approaches to shift testing left
- Keep sprint durations as short as possible (2-4 weeks)
- Consider using a practice sprint to determine mechanics of a sprint; delivering functionality is optional
- It takes a few sprints to get to working velocity; use them to get into a cadence and don't be too aggressive when planning initial sprints
- Include geographically distributed team members in ceremonies to enhance collaboration
- Mutually manage scope priorities with the client to understand the change impacts and tradeoffs (adding/removing user stories from roadmap as needed)
- Regularly perform PBR - always have at least 2-3 sprints worth of ready user stories
- Prepare testing execution before the testing phase begins – start early and allocate capacity to these preparation activities
- Prepare for deployment early
- Remember to infuse DevOps leading practices, such as –
  - All check-ins should be traceable to an user story or a defect
  - All developments and unit testing should be done locally
  - Consider multiple small-sized code commits (at least once a day)
  - Developers should collaborate on code in a single branch (called 'trunk') instead of creating other long-lived branches, , thereby avoiding merge and build issues
  - All the application code, configuration and static content (e.g., CSS, HTML) should be in the source control system
  - It is important to verify that the application runs as expected in a production-like environment before deploying it to production; this requirement should be added to the definition of "done" for the development.

Key tips

- Teams should release MVP sets as early and often as possible
- Practicing continuous delivery (CD) requires automation and management of the build, packaging, and deployment scripts, which can be addressed through version control check-ins
- Trunk-based development is a key enabler of continuous integration (CI) and, by extension, continuous delivery (CD)
- Any build should be deployable to any of the environments by ensuring that the source code has no environment-specific configurations
- The success of agile delivery relies on the ability to maintain the codebase in a deployable state at all times. Hence, the unit test suite should be run as part of the build process
- Services as a deployable unit should be mutually exclusive; independently deployable, testable, and scalable; and isolated from failures of other upstream or downstream application/services
- To achieve the DevOps outcomes, shift reliance to control mechanisms (like automated testing, automated deployment and peer review of code changes) to drive transparency, responsibility, and accountability between developers and operations
- Ensure monitoring alerts i.e. server monitoring and application performance monitoring are in place

Agile Quick Reference Guide

Deloitte.

Scrum team roles

Development team

- **Cross-functional** (with all skills necessary to create a product increment)**, self-organizing and self-managing team**, about 3-7 members
- Individual team members may have specialized skills/areas of focus, but accountability belongs to the team as a whole
- Committed to the work full time, not splitting their time over numerous projects or teams
- Perform the tasks of delivering the product increment
- Team maintains the sprint backlog, defines tasks and assignments
- Participates in product backlog refinement (PBR) to help update, groom, and estimate user stories for future sprints

Product owner (PO)

- **Voice of the customer:** regularly and consistently engages with business stakeholders and is accountable for product success
- One person per scrum team – not a committee
- Must be experienced, empowered, influential, committed, and trained
- Accountable for the product backlog, priorities, and defines all product features
- Ensures team is working on highest value features
- Negotiates work with the team by discussing the priorities along with the team's capacity and velocity

Scrum master (SM)

- A **servant leader** who enables close cooperation across all roles and functions
- Ensures that the team is fully functional and productive
- Shields and protects the team from external interferences
- Manages, removes, and escalates (as needed) impediments identified by the team

Artifacts

Product backlog

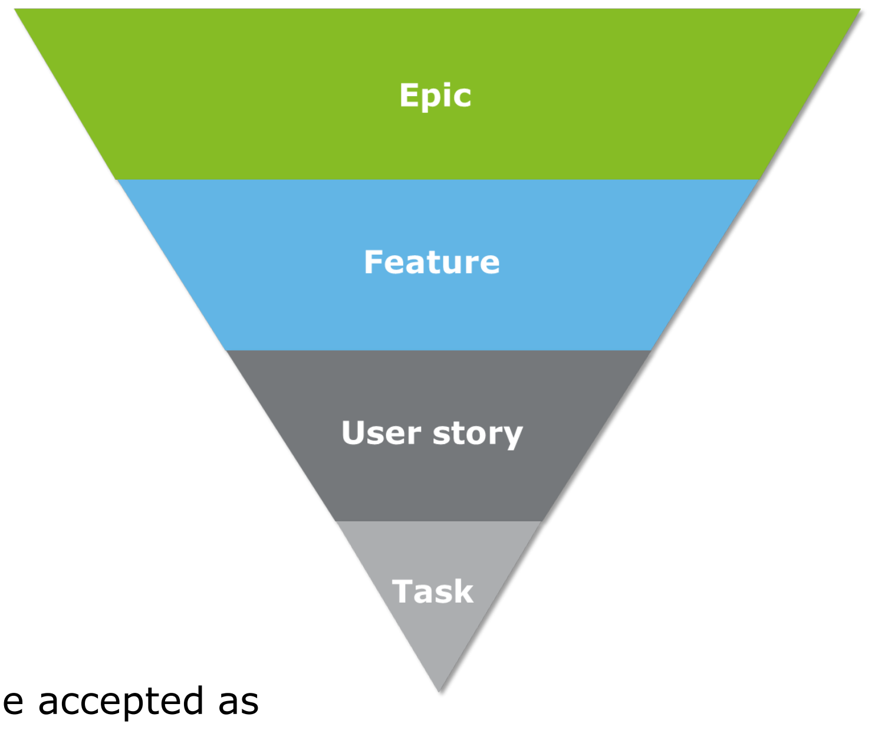
- A dynamic list, in priority sequence, of features and user stories to be developed
- Continually refined to contain at least 3 sprints' worth of ready user stories
- Prioritized and maintained by the PO

Sprint backlog

- A group of prioritized and estimated user stories decomposed into tasks that the development team commits to complete during the sprint
- It is created by the development team (based on direction of priorities set by the PO and sprint goals) during sprint planning and maintained by the development team during the sprint using story and task boards

User stories

- Descriptions of desired functionality, told from the perspective of the user
- Should not take more than half a sprint to meet the definition of done (DoD)
- Each story is captured as a separate item on the product backlog
- A user story includes a title, description (user story statement), acceptance criteria, and supporting documentation (e.g. wireframes, high-level design)
- **User story statement:**
  - **As a**…persona (who)
  - **I want to**…achieve a goal (what)
  - **So that**…tangible benefits that will be realized (why)
- **Acceptance criteria** explains the PO's conditions of satisfaction that the story must meet to be accepted as complete; the minimum functionality for a given story; allows the scrum team to gain a shared understanding of the complete story



Product roadmap

- Forecast of planned releases including epics, features, and user stories
- Medium to long-term planning is reflected in the roadmap and the product backlog
- Forecast is continually adjusted as priorities change and as sprints are completed

Product increment

- The sum of the user stories completed during a sprint
- The new product increment can be implemented by itself or packaged together with increments from previous sprints

Definition of ready (DoR)

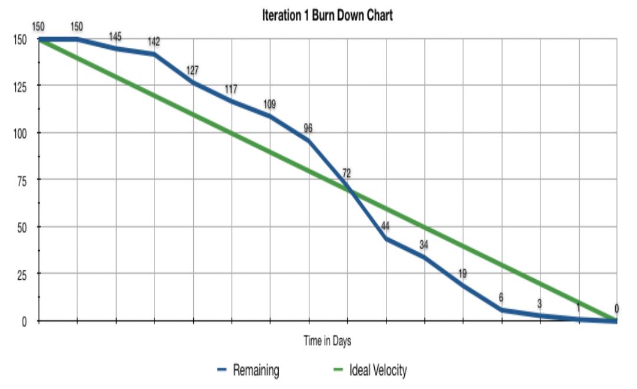
- Documents criteria any user story has to reach before the team can plan it in a sprint
- Applies to all user stories
- Ensures that the incoming work meets a basic level of quality, which will help prevent confusion and wasted time that the team might spend on trying to understand the requirements once the work begins
- Needs to be defined and understood for discovery and ongoing PBR

Definition of done (DoD)

- A quality checklist of activities that must be completed in order for any user story to be considered complete
- Applies to all user stories
- Improve and expand the DoD over time to ensure quality

Burndown chart

- Used to track and forecast progress
- Graphical representation of estimated work remaining in a sprint over time
- Calculated in story points or hours
- When calculated in story points, no partial credit – story points recognized only when stories are complete and meet DoD
- Generated by tool and monitored daily



Impediment list

- Items preventing team from completing work
- Maintained by SM
- Impediments can be removed upon resolution or during the daily standup by the team
- SM accountable for removing all impediments and escalating as appropriate

Ceremonies & events

Sprint

- A time-boxed iteration of one month or less during which scrum team(s) work to complete a set amount of work to create a "Done", useable, and potentially releasable product increment
- Sprints include sprint planning, daily stand-ups, development work and testing, sprint review, and sprint retrospective

Sprint planning

- Collaborative team working session to plan sprint; answers what can be delivered and how the work will get done
- Conducted on day 1 of each sprint, typically 2-4 hours for a 2-week sprint (scaled depending on sprint duration)
- Entire scrum team participates; SM acts as facilitator
- Velocity for sprint set based on confirming team capacity and reviewing historical velocity
- Sprint goals are defined and team selects user stories from product backlog for the sprint backlog based on priority (only "Ready" stories that meet DoR are considered) until team's velocity is met
- PO provides clarification on selected user stories
- Selected user stories are broken into tasks and development team estimates effort to complete tasks (in hours), considering DoD
- Development team commits that all stories can be completed as part of the sprint

Daily stand-up

- A daily 15-minute ceremony for the development team to optimize team collaboration and performance, focusing on progress toward the sprint goal (SM also attends, PO optional)
- Held every day (except for the first and last days of sprint) at the same time and place
- Development team members share progress with team (not to SM or PO), answering:
  - What did I do in the last 24 hours?
  - What will I do in the next 24 hours?
  - Any impediments/blockers in my work?
- Team often meets immediately after for detailed discussions regarding impediments (with PO)

Sprint review

- Held at the end of the sprint to share the outcomes of the sprint, elicit feedback, and foster collaboration
- Attended by entire scrum team, business owners, and stakeholders (often managers/execs)
- Typically 1-2 hours (no longer than 4 hours)
- Agile team demos "Done" user stories previously approved by PO (it is not an acceptance meeting)
- Feedback is gathered and owned by the PO to incorporate into product backlog

Sprint retrospective

- A closed team ceremony to identify continuous improvement opportunities to be implemented in the next sprint
- Typically 1 hour
- Held after the sprint review and prior to the next sprint planning
- Questions:
  - What did we do well?
  - What did not go well?
  - What can be improved?

Product backlog refinement (PBR)

- The act of adding detail, estimates, and order to items in the product backlog
- An on-going activity to ensure the product backlog is always being refined ahead of the teams who will pull work from it
- SMEs may spend a significant amount of time working on developing features and user stories → allocate appropriate capacity for PBR
- Before sprint planning, the entire team should have previewed and estimated user stories
- Schedule PBR sessions regularly with sufficient time to maintain backlog of 3 sprints worth of ready user stories (1-2 per week for 2 hours)

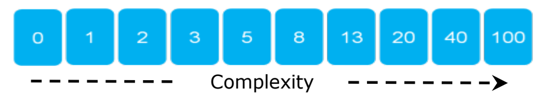
Scrum of scrums

- A technique to scale scrum up to large groups where representatives from clusters of scrum teams discuss cross-team integrations, dependencies, impediments
- Also used for process reminders/changes to ensure hygiene or make improvements
- Attended by SMs, POs, select development team members
- Plan for 2-3 scrum of scrums per week

Other terms & techniques

Relative estimation

- Estimate features using t-shirt sizing, user stories in story points, and tasks in hours
- T-shirt sizing (XS, S, M, L, XL) is an effective way of removing the element of time from the level of effort
- Story point estimating is done during PBR as the user story is further detailed. Story points represent the relative measure of the size or complexity of a story, based on a modified Fibonacci sequence; collaborative estimation techniques like planning poker drive discussion and consensus on the work that needs to be done



- Task estimation in hours is done during sprint planning by the team committing to delivering it and is based upon the actual effort each task is expected to consume
- Develop estimation guidelines for scrum teams so that estimates are generally consistent from one team to the next

Persona

- A generalized character that represents a set of user types that will be using the solution in a similar way
- Represents how the specific persona interacts with the system so that the solution features meet user needs and expectations

Team capacity

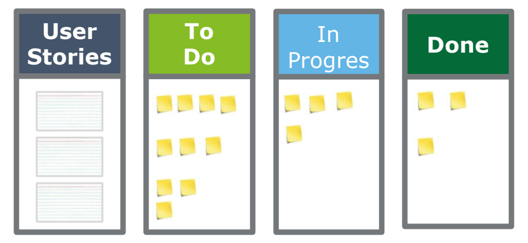
- Total number of hours each team member is available to work on backlog items during the upcoming sprint
- Initially calculated during discovery and revisited in sprint planning
- Considers holidays, PTO, training, conferences, etc.
- Capacity = total FTE days in the sprint x ideal hours
- Ideal hours guideline is ~6 hours/day to allow time for meetings, overhead, non-sprint work

Team velocity

- The estimated total number of story points an scrum team can be expected to complete in a sprint
- Calculated as the rolling average of the velocity from the previous 3 sprints
- Used to plan for sprint
- Velocity should not be compared across teams

Story & task boards

- Used to visualize and manage the work
- Includes all the user stories and tasks for the sprint
- Each item's progress is tracked by the team during sprint execution
- Impediments can be tagged against the board
- Can be physical and/or accessed in tool



Story mapping

- A visual representation of a product backlog to make end-to-end product scope visible
- An arrangement of user stories in a two-dimensional map resulting in a sequential narrative from left to right and priorities from left to right

Minimum viable product (MVP)

- The smallest subset of features that can be implemented together to deliver value to the business
- Scope is sliced horizontally across the story map to deliver value

Agile Quick Reference Guide

Deloitte.

Manifesto for agile software development

The agile manifesto states: 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 and interactions	over	*Process and tools*
Working software	over	*Comprehensive documentation*
Customer collaboration	over	*Contract negotiation*
Responding to change	over	*Following a plan*

While there is value in the items on the right,
we value the items on the left more. – agilemanifesto.org

Iterative

An **iterative** process is one that makes progress through successive refinement. The first iteration of a product may be incomplete or weak in some areas. The team then iteratively refines those areas until the product is satisfactory. With each iteration, the product is improved through the addition of greater detail.

Incremental

An **incremental** process is one in which a product is built and delivered in pieces. Each piece, or increment, represents a complete subset of functionality. The increment may be either small or large. New increments are created until the product is complete

Scrum and agile are both incremental and iterative. They are iterative in that they plan for the work of one iteration to be improved upon in subsequent iterations. They are incremental because completed work is delivered throughout the project.

Mountaingoatsoftware.com

Empirical behaviors

em·pir·i·cal
/əmˈpirik(ə)l/

based on, concerned with, or verifiable by observation or experience rather than theory or pure logic.

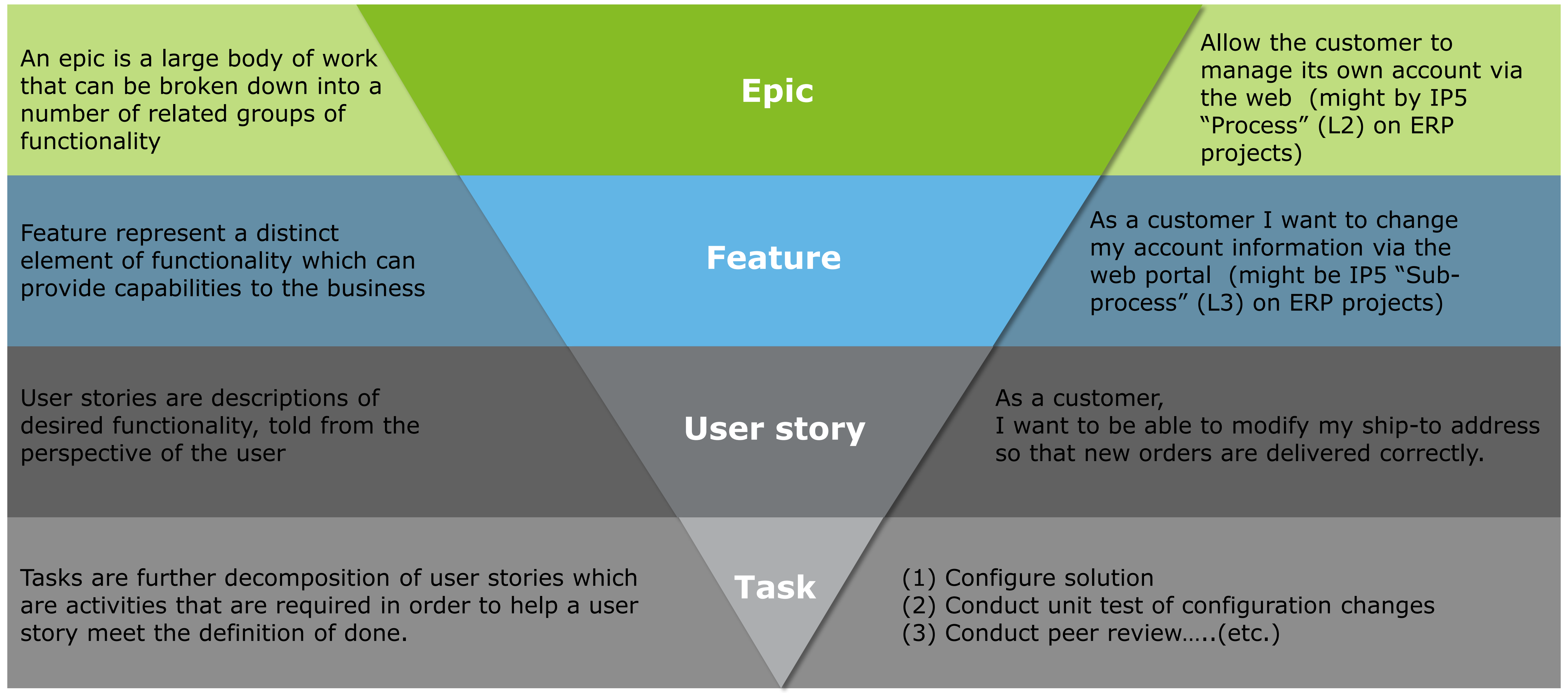
Decisions are made based on observation and experimentation rather than on detailed upfront planning. Empirical process control relies on the three main ideas of transparency, inspection, and adaptation.

Scrumstudy.com

Agile guiding principles

Satisfy the customer	Our highest priority is to satisfy the customer through **early and continuous delivery of valuable software**.	**Working software is primary**	**Working software** is the primary measure of progress.
Welcome Change	**Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.	**Maintain a constant pace**	Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
Deliver frequently	**Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	**Focus on technical excellence**	**Continuous attention to technical excellence and good design** enhances agility.
Work with business people	Business people and developers must **work together daily throughout the project**.	**Keep it simple**	**Simplicity--**the art of maximizing the amount of work not done--is essential.
Support motivated people	**Build projects around motivated individuals**. Give them the environment and support they need, and trust them to get the job done.	**Teams should self-organize**	The best architectures, requirements, and designs emerge from **self-organizing teams**.
Use face-to-face conversation	The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.	**Reflect and adjust**	At regular intervals, the **team reflects on how to become more effective,** then tunes and adjusts its behavior accordingly.

Decomposing business requirements



An epic is a large body of work that can be broken down into a number of related groups of functionality

Epic

Allow the customer to manage its own account via the web (might by IP5 "Process" (L2) on ERP projects)

Feature represent a distinct element of functionality which can provide capabilities to the business

Feature

As a customer I want to change my account information via the web portal (might be IP5 "Sub-process" (L3) on ERP projects)

User stories are descriptions of desired functionality, told from the perspective of the user

User story

As a customer,
I want to be able to modify my ship-to address so that new orders are delivered correctly.

Tasks are further decomposition of user stories which are activities that are required in order to help a user story meet the definition of done.

Task

(1) Configure solution
(2) Conduct unit test of configuration changes
(3) Conduct peer review…..(etc.)

Epics and features are simply ways of grouping user stories into a hierarchy so that they can be discussed simply – don't get hung-up on rigid definitions.