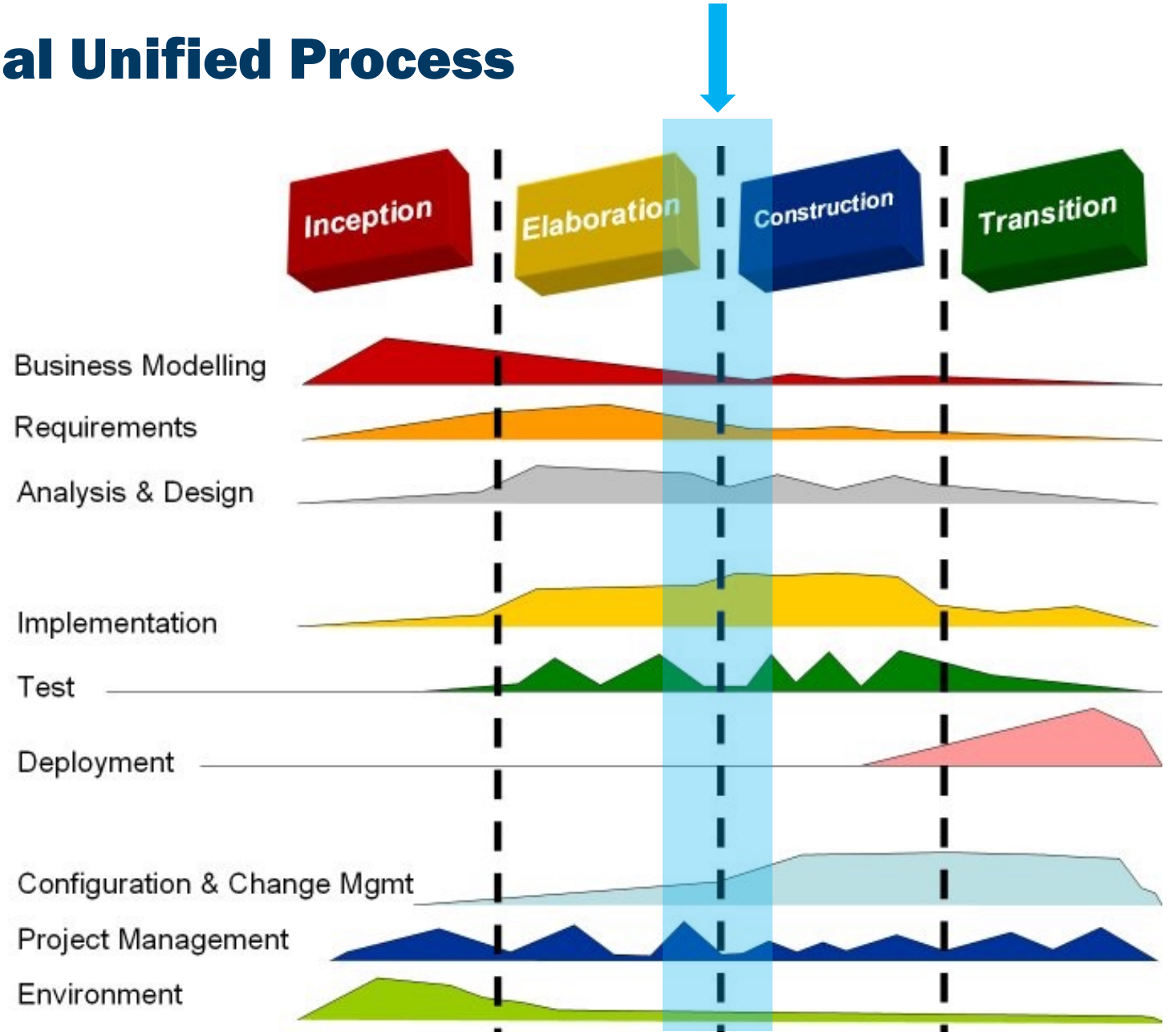
# Software Project Management

L.EIC-ES-2021-22

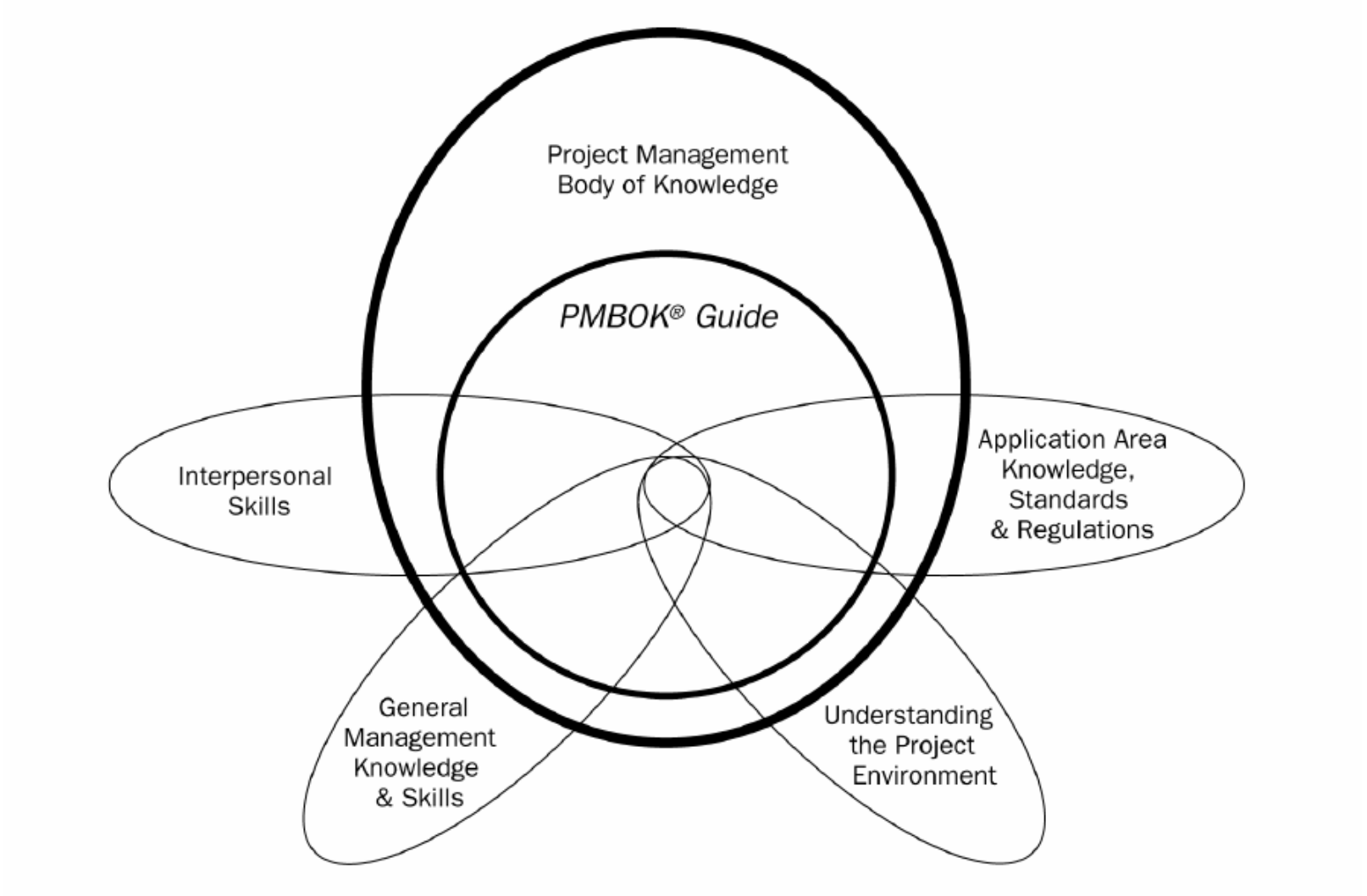
**Ademar Aguiar, J. Pascoal Faria**

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

# Rational Unified Process

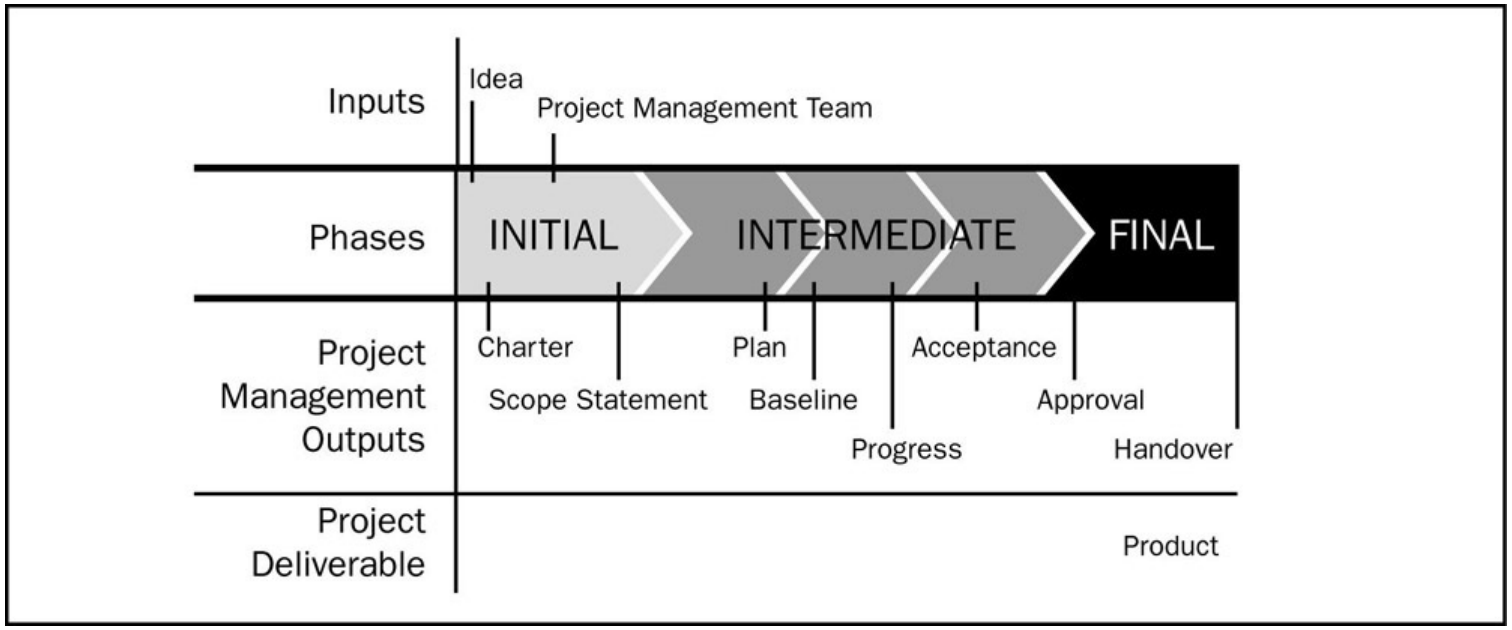


# Areas of Expertise

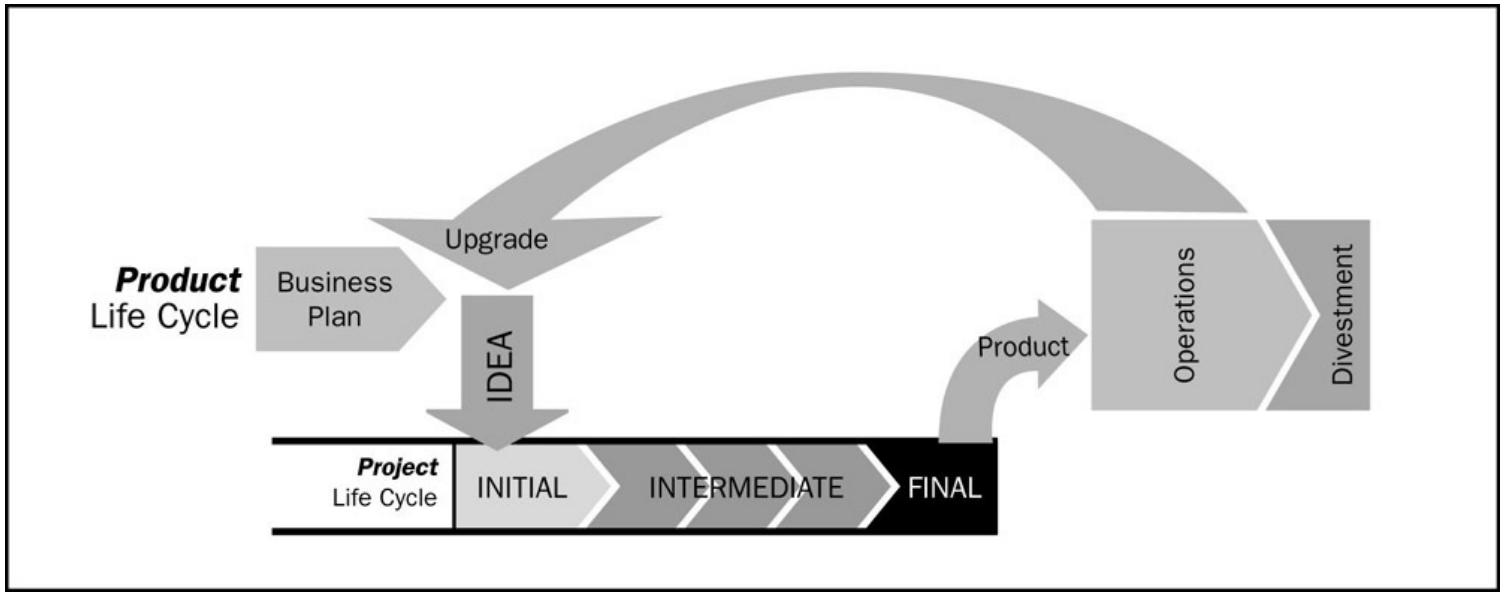



From "*A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* Third Edition"

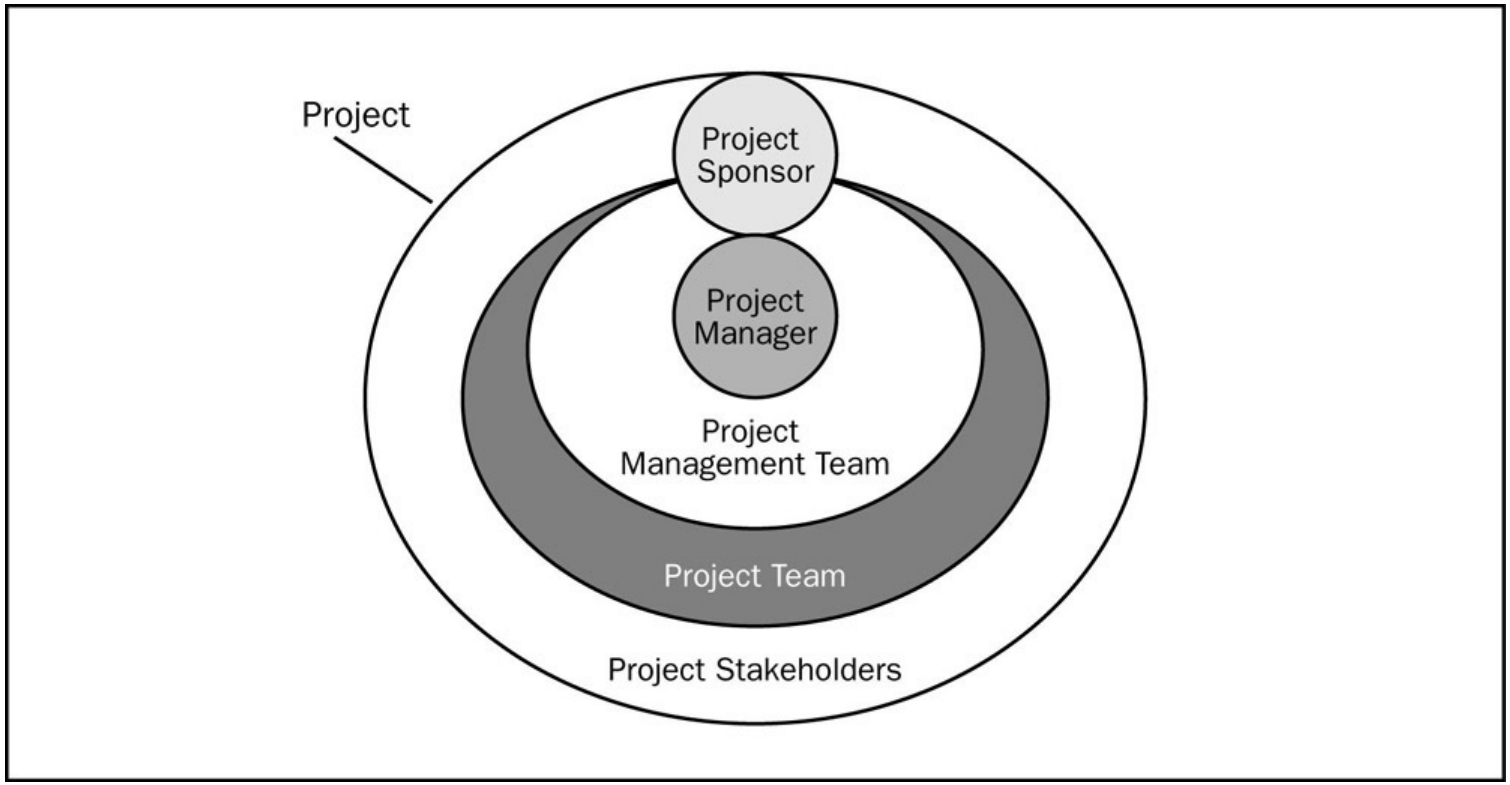
# Typical Sequence of Phases



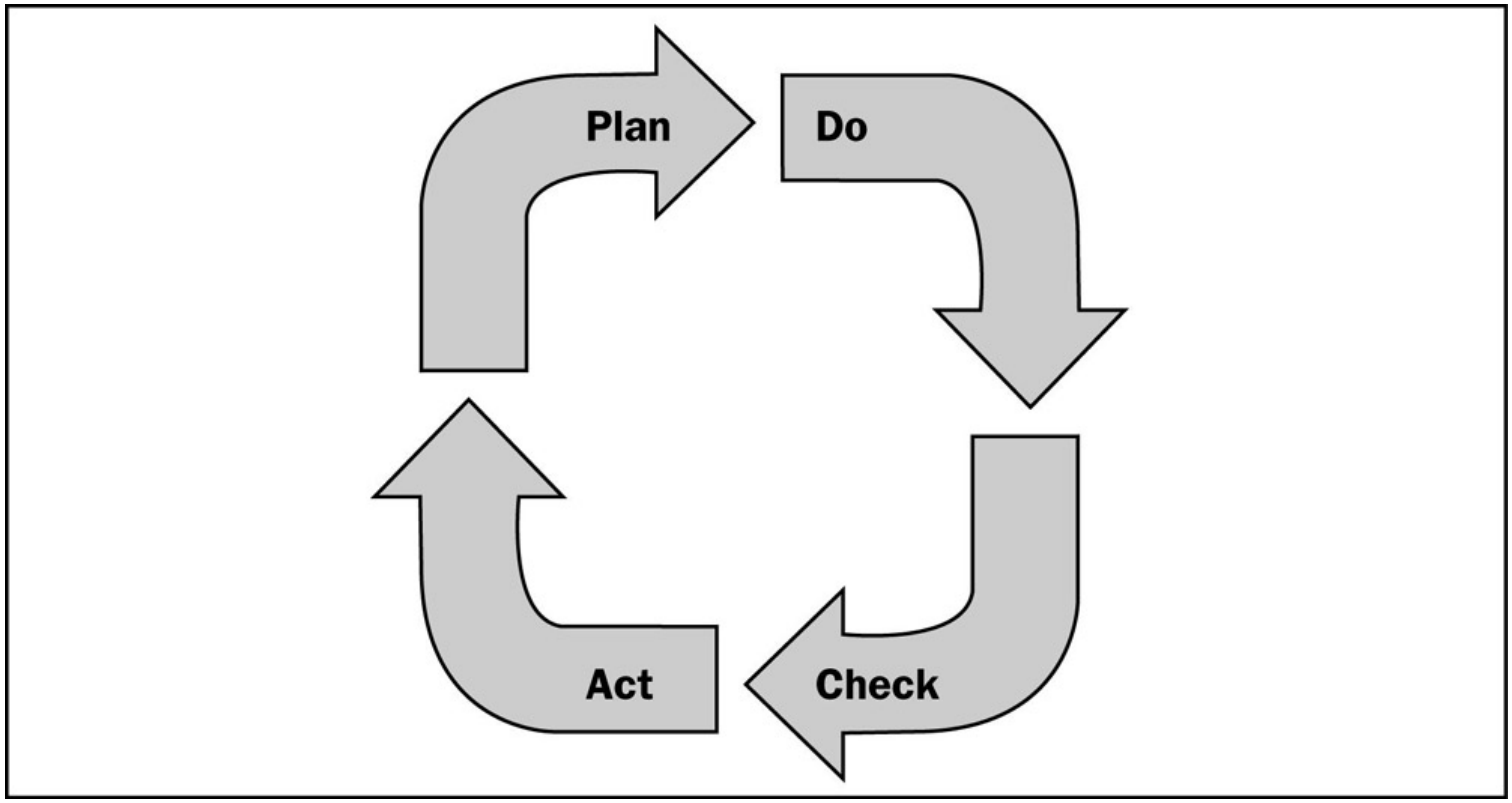
# Product and the Project Life Cycles



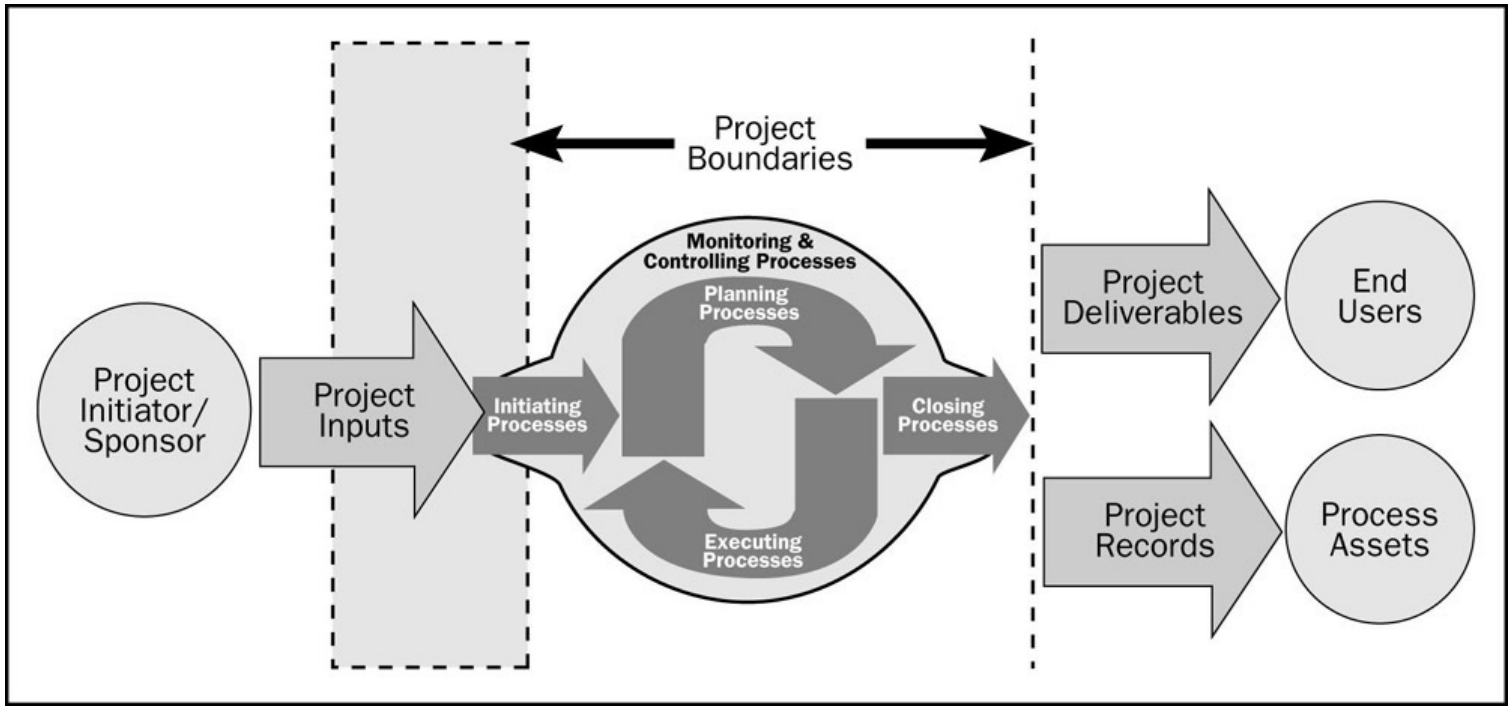
# Project Stakeholders



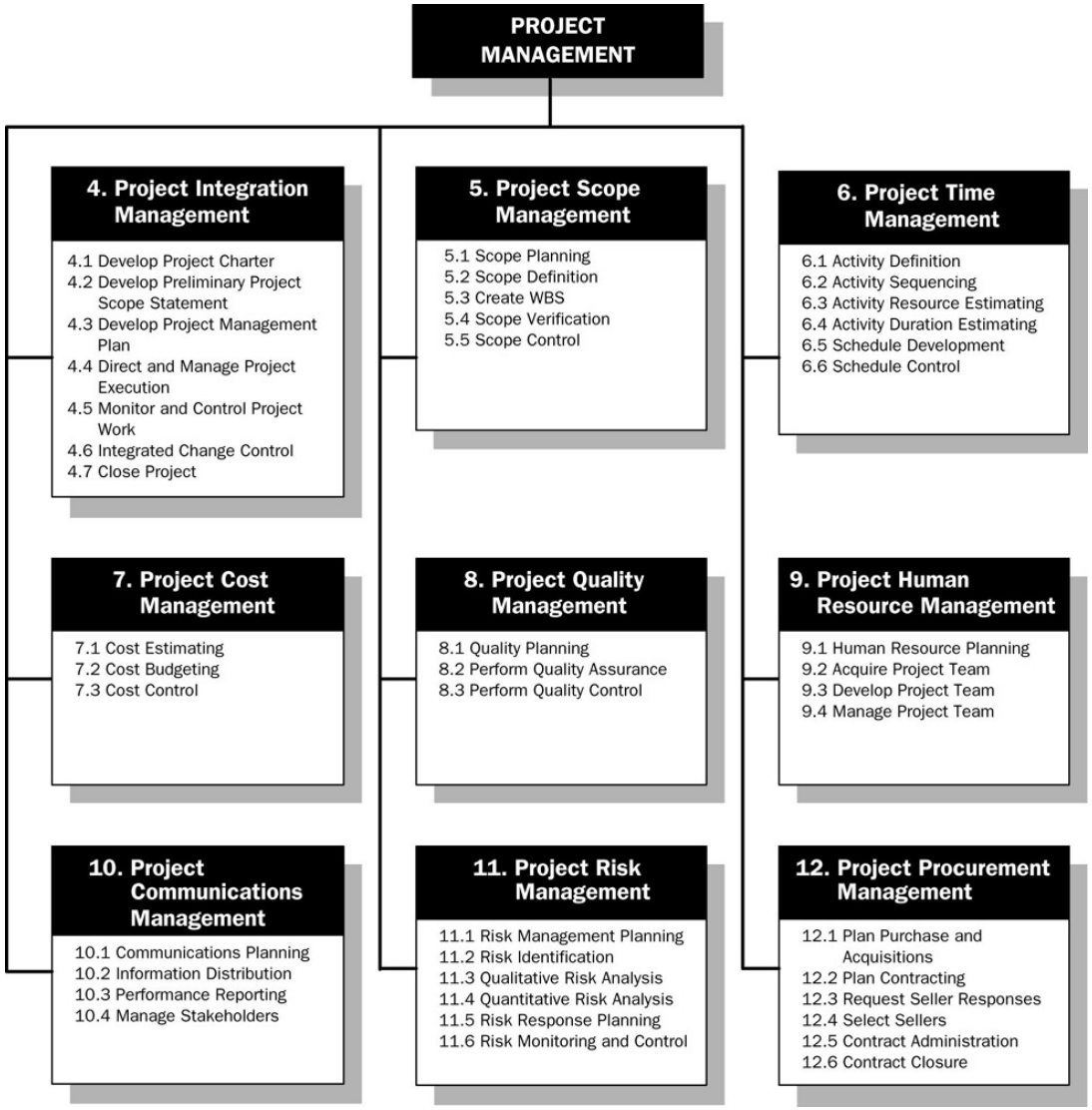
# Plan-Do-Check-Act Cycle



# Project Boundaries

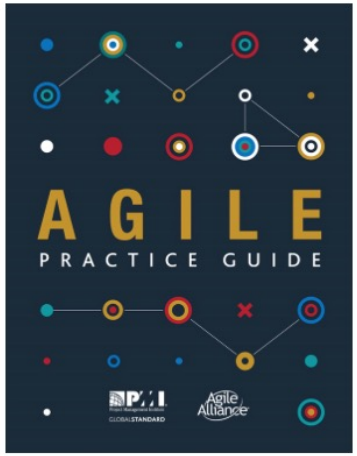


# PMBOK Guide


PROJECT MANAGEMENT
4. Project Integration Management
4.1 Develop Project Charter
4.2 Develop Preliminary Project Scope Statement
4.3 Develop Project Management Plan
4.4 Direct and Manage Project Execution
4.5 Monitor and Control Project Work
4.6 Integrated Change Control
4.7 Close Project
5. Project Scope Management
5.1 Scope Planning
5.2 Scope Definition
5.3 Create WBS
5.4 Scope Verification
5.5 Scope Control
6. Project Time Management
6.1 Activity Definition
6.2 Activity Sequencing
6.3 Activity Resource Estimating
6.4 Activity Duration Estimating
6.5 Schedule Development
6.6 Schedule Control
7. Project Cost Management
7.1 Cost Estimating
7.2 Cost Budgeting
7.3 Cost Control
8. Project Quality Management
8.1 Quality Planning
8.2 Perform Quality Assurance
8.3 Perform Quality Control
9. Project Human Resource Management
9.1 Human Resource Planning
9.2 Acquire Project Team
9.3 Develop Project Team
9.4 Manage Project Team
10. Project Communications Management
10.1 Communications Planning
10.2 Information Distribution
10.3 Performance Reporting
10.4 Manage Stakeholders
11. Project Risk Management
11.1 Risk Management Planning
11.2 Risk Identification
11.3 Qualitative Risk Analysis
11.4 Quantitative Risk Analysis
11.5 Risk Response Planning
11.6 Risk Monitoring and Control
12. Project Procurement Management
12.1 Plan Purchase and Acquisitions
12.2 Plan Contracting
12.3 Request Seller Responses
12.4 Select Sellers
12.5 Contract Administration
12.6 Contract Closure

# Agile Practice Guide






**PRACTICE GUIDE** | Agile Practices | September 2017

**How to cite this article:**
Agile Practice Guide (2017).

Receive the *Agile Practice Guide* when you purchase the *PMBOK® Guide* – Sixth Edition.

ORDER NOW

Created in partnership with Agile Alliance®, the *Agile Practice Guide** provides tools, situational guidelines and an understanding of the various agile approaches available to enable better results. It is especially useful for those project managers accustomed to a more traditional environment to adapt to a more agile approach.

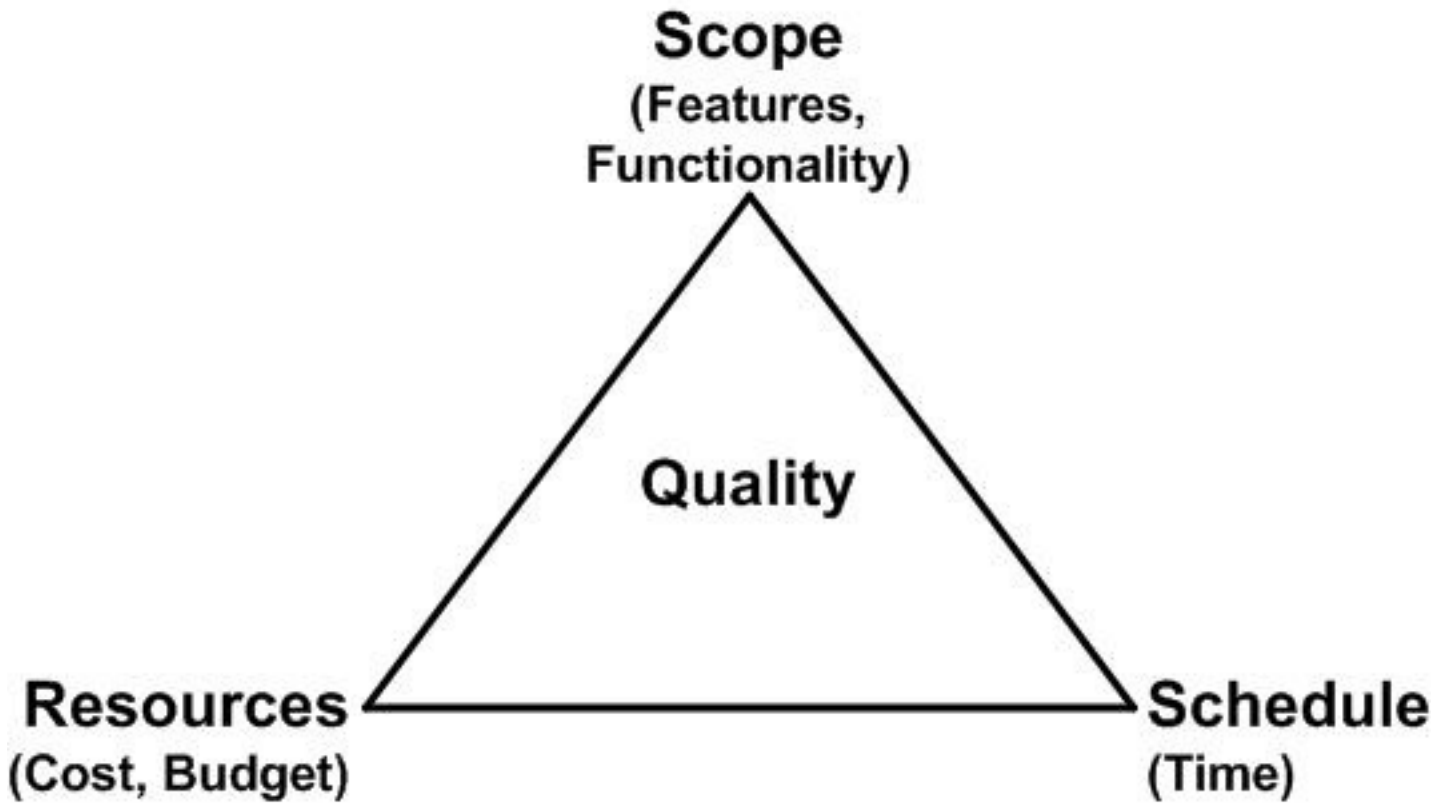
The *Agile Practice Guide* contains the following sections:

- **An Introduction to Agile** describes the *Agile Manifesto* mindset, values and principles. It also covers the concepts of definable and high-uncertainty work, and the correlation between lean, the Kanban Method and agile approaches.
- **Life Cycle Selection** introduces the various life cycles discussed in the practice guide and covers suitability filters, tailoring guidelines and common combinations of approaches.
- **Implementing Agile: Creating an Agile Environment** talks about critical factors to consider when creating an agile environment such as servant leadership and team composition.

## Errata Sheets

Find the latest corrections and updates to the Agile Practice Guide

# The Iron Triangle



# Controlling Software Projects

- Four control variables require balancing
  - Resources
  - Time
  - Scope
  - Quality
- It is not advisable to set a priori the value of all variables simultaneously, if we want a successful project.
- "*scope*" is often the most important variable to control

# The Resource Variable

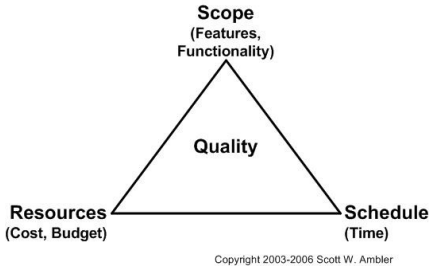
- Staffing is usually the **least effective** variable to adjust.
  - Staffing increases have long lead times.
  - Increased intensity has diminishing returns.
  - Team culture requires some degree of stability.
- Tools and technology can provide benefits.
  - Effective tools provide continuing benefits.
  - Front-end costs need to be carefully amortized.
  - The wrong tools and technology increase friction.

# The Time Variable

- Can be the **most painful** variable to adjust
  - Early commitments are usually date-based.
  - Target dates are often the most important objective.
  - Within a date boundary, there's only so much time.

# The Scope Variable

- Can be the **most effective** variable to adjust
  - Can adjust scope breadth – what's included.
  - Can adjust scope depth – refinement.
  - Partial scope can often generate immediate returns.
  - It is often preferable to reach a date with partial scope completely finished, rather than complete scope partially finished.



# Differences on Managing Agile Processes

- Iterative and incremental
- Parallel and concurrent, not phased
- Planned around deliverables, not activities
- Dynamic project balancing via scope adjustments
- Heavy emphasis on collaboration
- Management by facilitation

# Iterative and Incremental

- Iterative
  - Repeatedly executing nested process cycles
  - Iterations provide synchronizing points
  - Iterations provide feedback points
- Incremental
  - System is built in progressive stages
  - Iterations add features and refinements
  - Increments are working systems

# Paralell and Concurrent Activities

- Phased Approach
  - Gathers similar activity types together
  - Preference towards serial completion
  - Ultimate in phased approach is waterfall
- Concurrent and Parallel
  - Activities occur opportunistically
  - Activities of all types happening at the same time
  - Partial completion considered the norm

# Predictive vs. Agile Planning

- Predictive Planning
  - Creation of comprehensive activity-based plans
  - Execution of defined activities to follow plan
  - Management by controlling activities to conform to plan
- Agile Planning
  - Creation of prioritized set of deliverables
  - Opportunistic execution of activities to create deliverables
  - Management via feedback and adaptation

# Project Balance in an Agile Process

- Sustainable resource management
  - Stable teams
  - Steady pace
  - Favor high ROI tools and technology
- Fixed time management
  - Time-boxed development cycles
- Adaptive scope management
  - Feedback-based scope adjustments

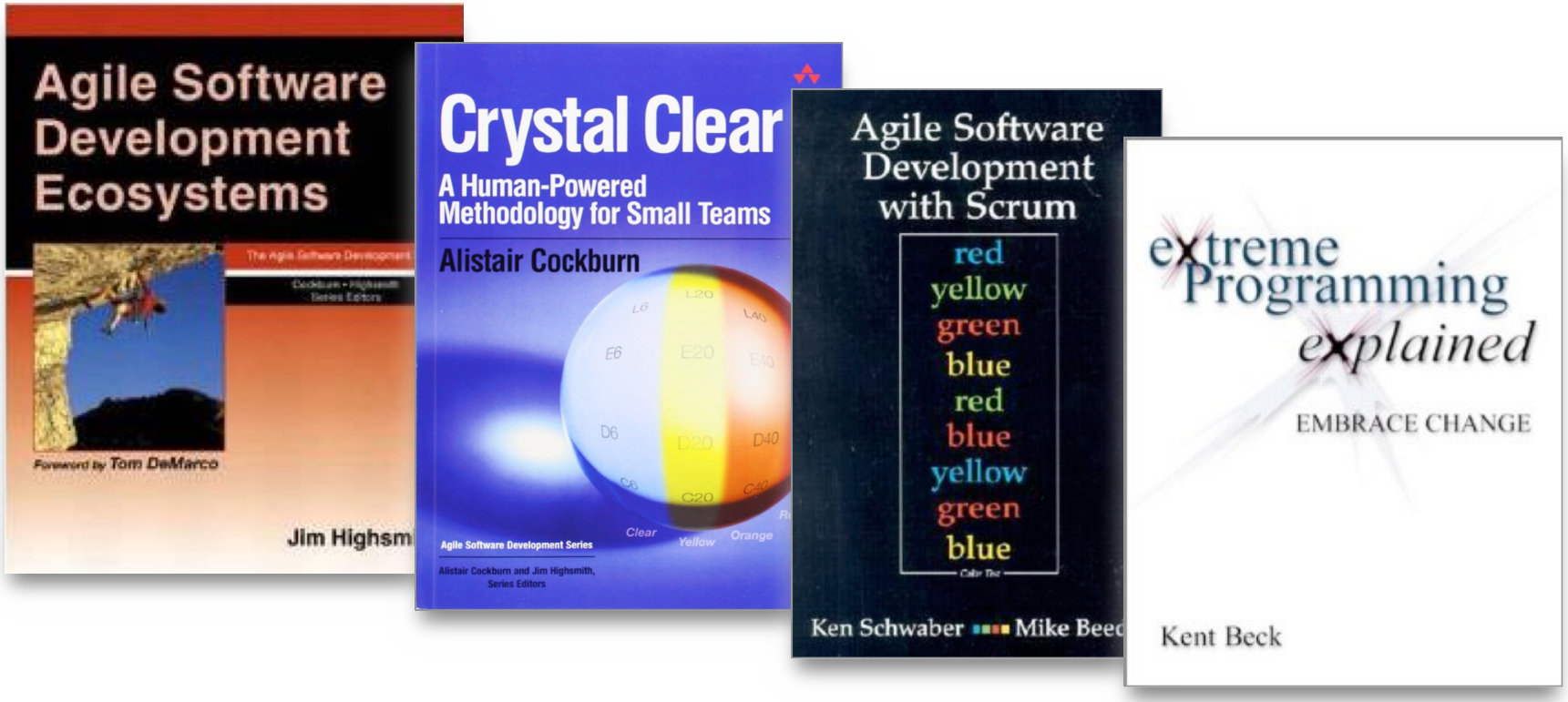
# “Heroic” vs. “Collaborative”

- Heroic development emphasizes individuals
  - Activities assigned to individuals
  - Project results heavily dependent on individual performance
  - Increases “keyhole” risks
- Collaborative development emphasizes teams
  - Teams self-organize activities to meet goals
  - Teams leverage diverse skills
  - Teams mitigate keyhole risks

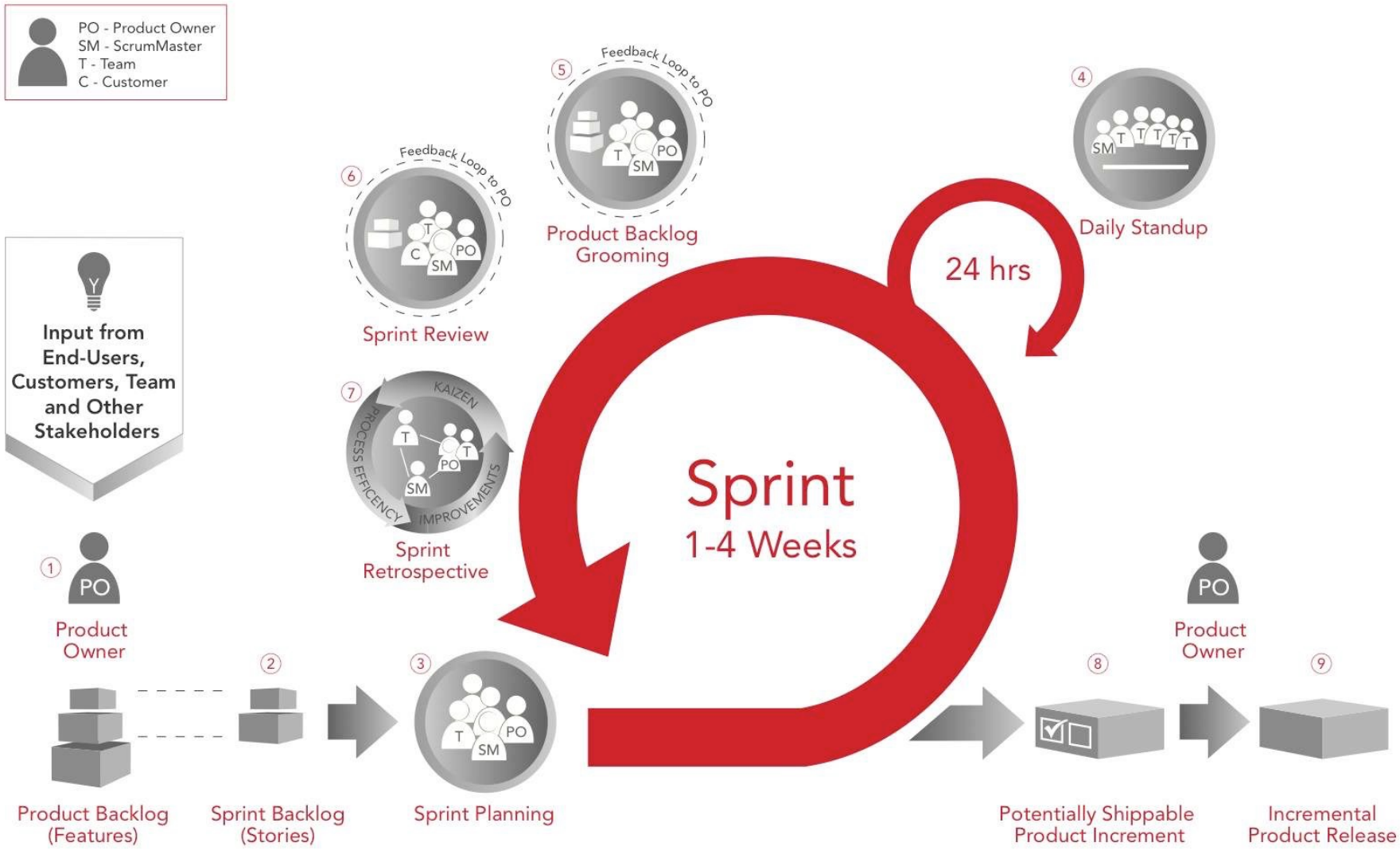
# Management by Facilitation

- Traditional “Command and Control Strategy”
  - Decisions made by central authorities
  - Activities delegated
  - Manager controls activities
- Replaced by “Facilitation and Empowerment Strategy”
  - Decisions made by those with the most info
  - Activities accepted
  - Team self-manages and adapts
  - Organization ensures supportive environment

# Agile: ASD, Crystal Clear, Scrum, XP...

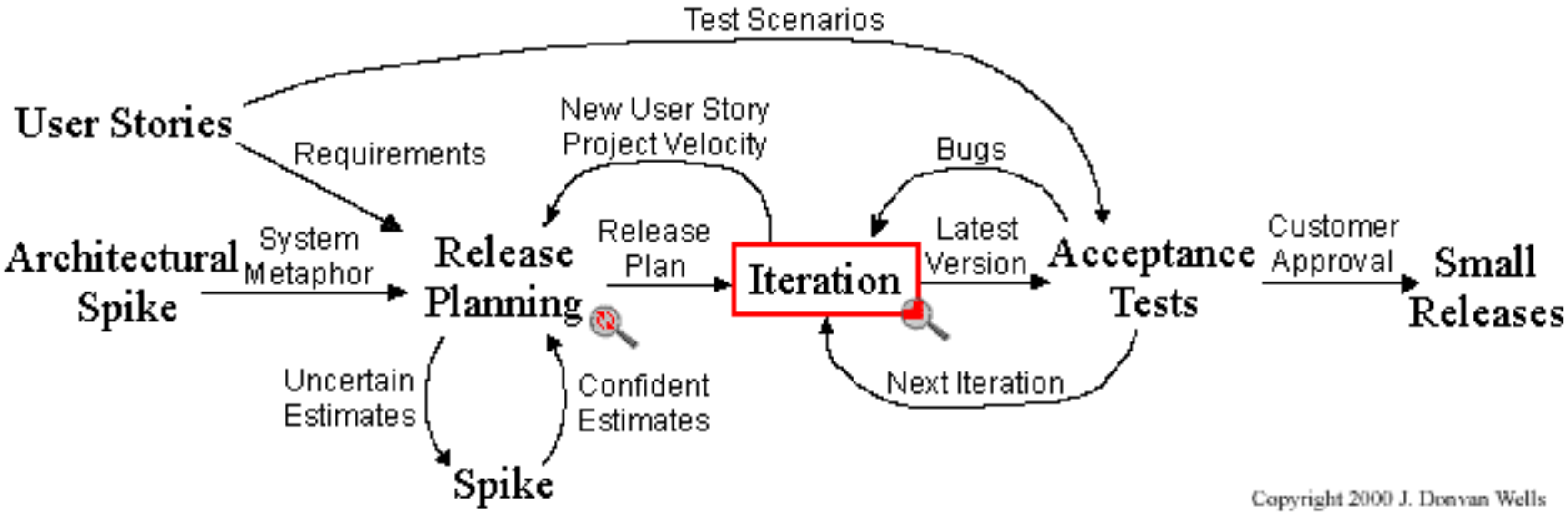


# How Scrum Works

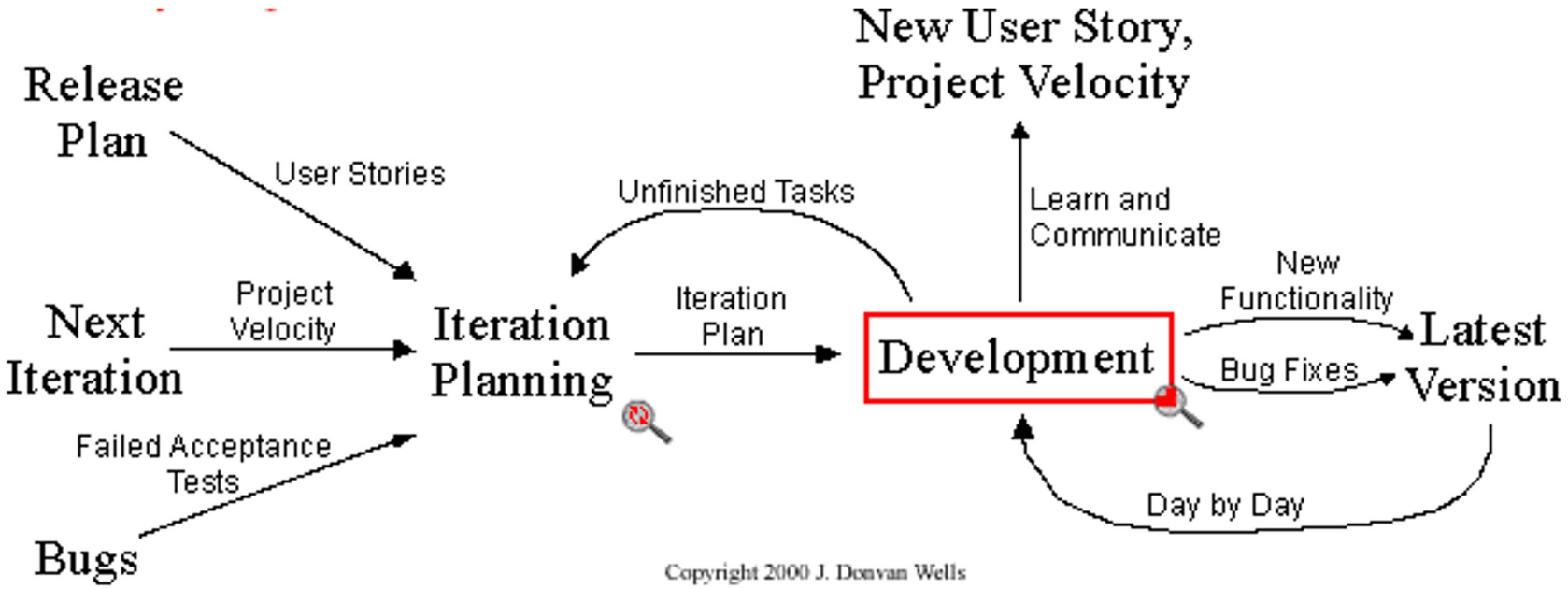



scruminc.

# Extreme Programming



# Extreme Programming




**Release planning: Release content & date**

# Iteration 0

- To plan or not plan?
  - Some people think agile development gives developers license to dive in and build, spending little or no time on early requirements gathering or architectural issues.
  - Projects in which months and months of planning, requirements specification, and architectural philosophizing occur before they deliver any customer value are not positively evaluated.
  - Iteration 0 is a practice that can help teams find that middle ground and balance anticipation with adaptation.
- The meaning of "0"
  - Nothing useful to the customer—features, in other words—gets delivered in this time period.
  - However, the fact that we have designated an iteration implies that the work is useful to the project team: architecture work, technological training, requirements document to base a contract signing,

# Iterations 1–N

- Iteration planning
  - assigns features to iterations for the duration of the project helps getting a feel for the project flow and determination of completion dates, staffing, costs, and other project planning information
- Activities of iteration planning
  - Determining how identified risks will influence iteration planning
  - Identifying the schedule target
  - Establishing the milestone and iteration periods
  - Developing a theme for each iteration (or milestone)
  - Assigning feature cards to each iteration, balancing customer priorities, risks, resources, and dependencies as necessary
  - Summarizing the plan in some combination of a feature-level spreadsheet plan, a feature card layout (usually on a wall), or a project parking lot
  - Calculating an initial project schedule from staff availability and total feature effort estimates
  - Adjusting the completed plan as necessary
- Increasing schedule reliability
  - +10% assigned to a “Rework and contingency” feature card
  - 1 or more “empty” iterations at the end of the project

# Three Types of Iteration Plans

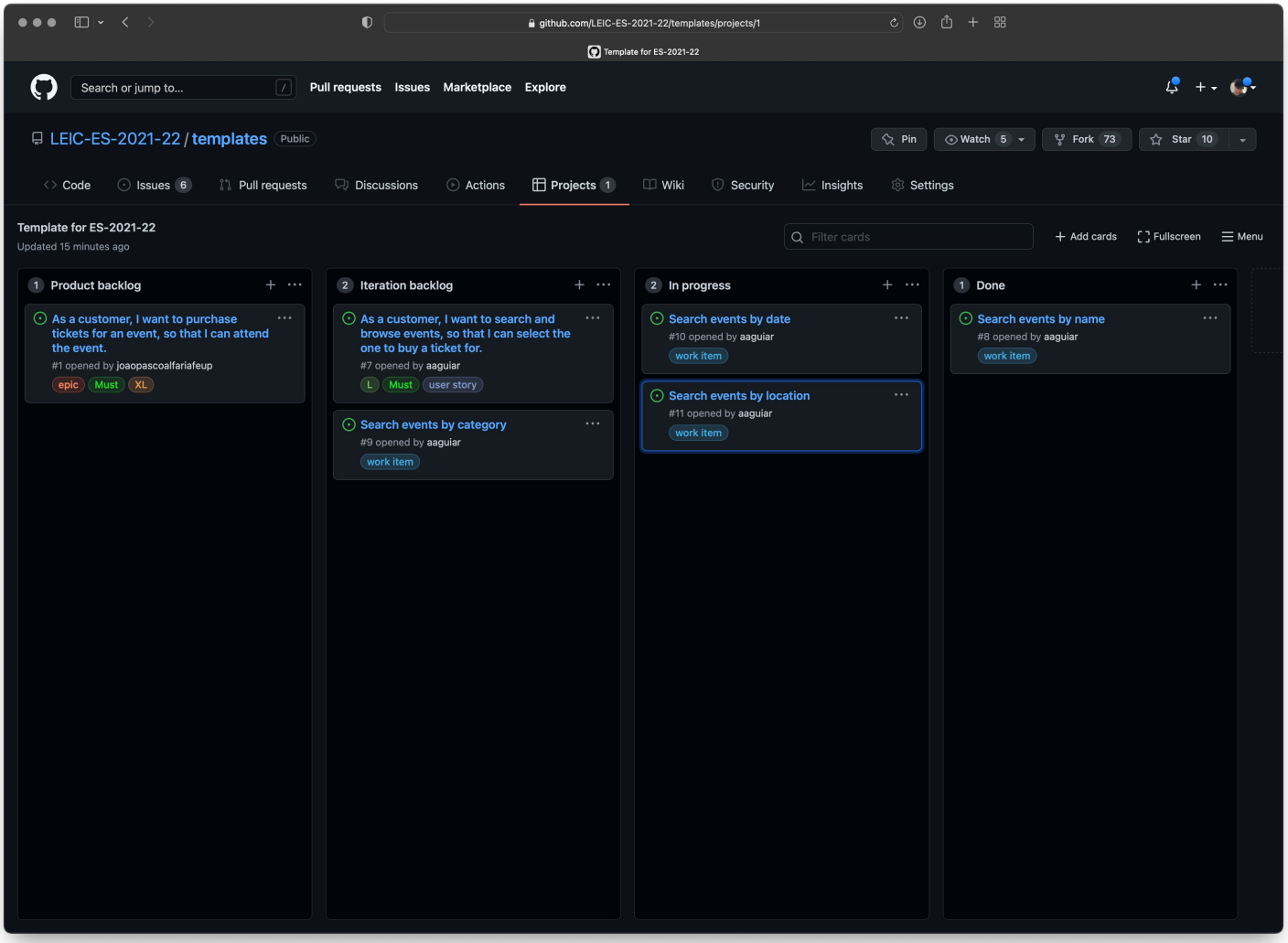
- Complete
  - A complete plan with features assigned to every iteration.
- Two-iteration plan
  - A two-iteration plan utilizing only a next iteration and then everything after
- One iteration
  - An iteration-by-iteration plan
- Best type of plan?
  - Depends on the nature of the project and the expectations of customers and stakeholders.
  - High exploration-factor projects suggest “one-iteration” plan that selects features for the first iteration, and continue only with a vague idea of the rest of the project.

# Next Iteration Plan

- Activities
  - To construct the list of activities to implement each feature, recording it on the back of the card.
- Reestimation
  - The team reestimates the work effort based on the more detailed assessment and adjusts the features planned for the iteration if necessary.
- Assignment
  - Team members sign up for features or activities based on their skills and/or desires.
  - Taking on the responsibility for getting the work done reinforces each individual's commitment to the project and thereby contributes to building a self-organizing team.

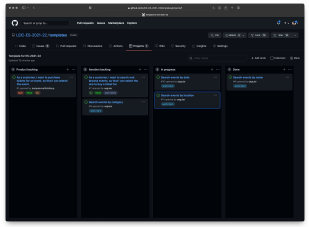
# Github Projects @ LEIC-ES-2021-22

# Github Projects



# Board & Columns

- Product backlog
  - an ordered list of items containing all epics, all user stories, all work items that are envisioned to be needed to be implemented to deliver the whole product
  - the order is always done from the perspective of the customer/end-user/stakeholder, based on value and effort, and is sometimes done by someone outside the team, or member of the team.
  - each item should be easy to understand by non-technical people and the team.
  - items are considered ready when have value and effort assigned.
- Iteration backlog
  - a subset of items (epics, user stories, work items, etc.) taken from the top of the Product backlog that, based on effort, are estimated to be able to be implemented during the iteration, and released at the end of the iteration.
  - items should be easy to understand by the team members.
- In progress
  - items being developed at the moment, and only at the moment.
  - Items have team members assigned.
  - Work In Progress (WIP) should be no more than the number of members, or pairs, of the team.
- Done
  - items already implemented, tested, and accepted to be valid and correct.



# User Stories

- Write the user stories in a template such as:

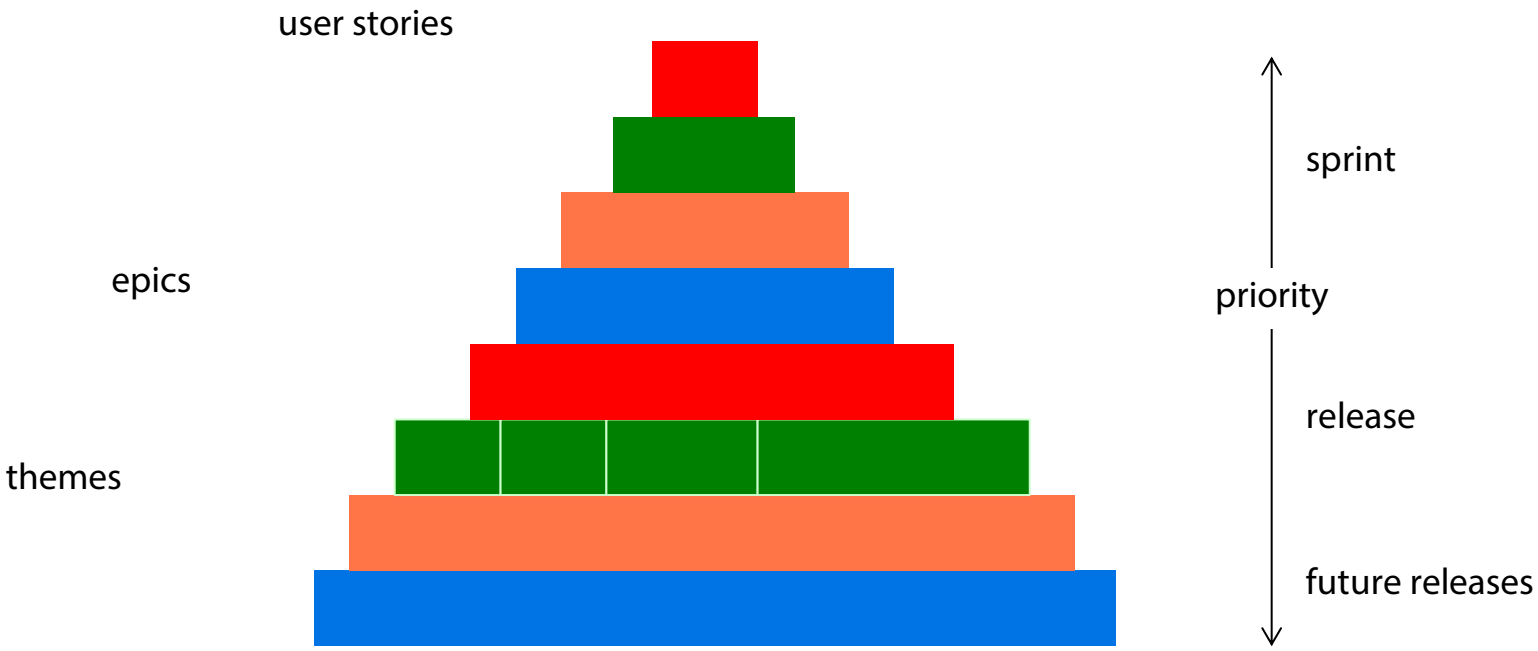
  As a <user role>,

  I want <goal>

  so that <reason>.

- **INVEST** in high quality user stories, i.e.:
  - Independent
  - Negotiable
  - Valuable
  - Estimable
  - Small
  - Testable

# Themes, epics, user stories -> priorities



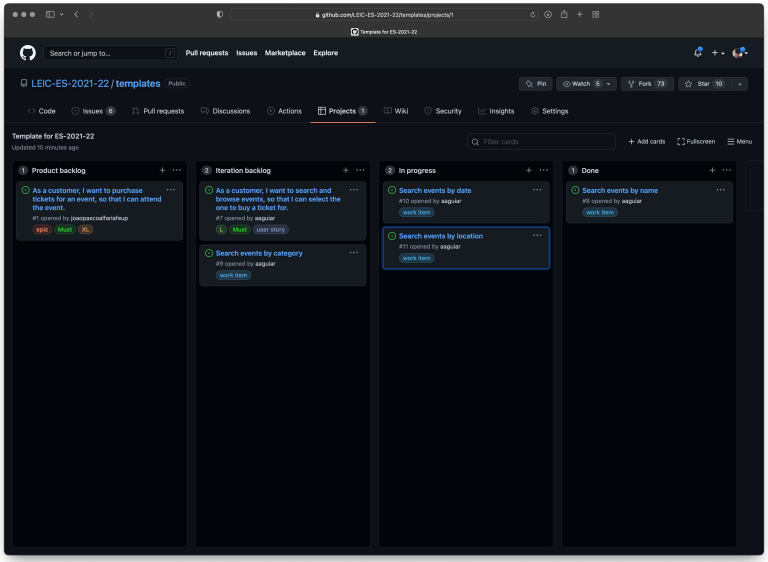
# Estimation

- How to estimate the unknown?
  - You can't!
  - When there are unknowns, you are guessing, not estimating.
  - Time and cost are often viewed as constraints, not estimates, in agile projects.
  - Agile organizations learn to live with uncertainty rather than trying to demand certainty in a fast-changing world.
- How to estimate by features rather than activity?
  - Estimate requirements gathering on a feature-by-feature basis, instead of for the overall project
- How to do progressive estimation?
  - Bottom up and top down, comparisons to similar projects, and using estimating tools, can help teams arrive at better overall project estimates, but they can't make up for uncertainty.
  - Multiple techniques can provide a better estimate for the entire project.
  - Team member estimates should be used for the next iteration plan.

# Value and Effort

- Value
  - we will use the MoSCoW method to assign value to each Product backlog item:
  - Must Have, Should Have, Could Have, Will Not Have (yet)
- Effort
  - we will use the t-shirt size method to assign an estimated effort to each Product backlog item.
  - XS, S, M, L, XL

Github: we will use labels to assign value and effort to items.



# Scope Evolution

- Reality is more complex than an admonition to "avoid scope creep" can handle.
- Some scope changes are inexpensive but valuable.
- Some scope changes are extensive and expensive but crucial to delivering customer value.
- In general, scope changes incorporated to meet evolving customer requirements and undertaken with an understanding, and approval, of their impact on the project increase the probability of project success.
- Agile development encourages change that arises from evolving knowledge, while at the same time it discourages the gold plating and requirements bloat that often occur in traditional up-front requirements gathering.
- Agile development is about focus and balance—focusing on the project's key vision and goals and forcing hard tradeoff decisions that bring balance to the product.
- Agile development plans by feature, in customer terminology, thereby concentrating the planning process on something the customer can relate to and prioritize easily.
- Because plans are adjusted each iteration based on actual development experience, not someone's guesses or wishes, nice-to-have features are pushed into later iterations and are often eliminated completely.
- A product's scope should be driven by customer value, technical feasibility and cost, the impact on a product's adaptability, and critical business schedule needs. It should not be held hostage to a plan developed when our product and project knowledge was still in its infancy.


ademar.aguiar@fe.up.pt