Applied Use Cases and Requirements Engineering

Michael Glienecke, PhD

Senior Principal Architect

Executive Director

What you know / have heard already

- Requirements
  - Functional / Non-Functional
  - Stakeholders
  - Ways to get them
- Use Cases
  - Actors, scenarios
  - UML
  - With Requirements
- Agile and Plan-Driven processes
  - Pros / Cons, implications

Use with caution

- Everything in here is my personal view
- You have a wonderful brain yourself, so take what you want and leave what you consider inappropriate

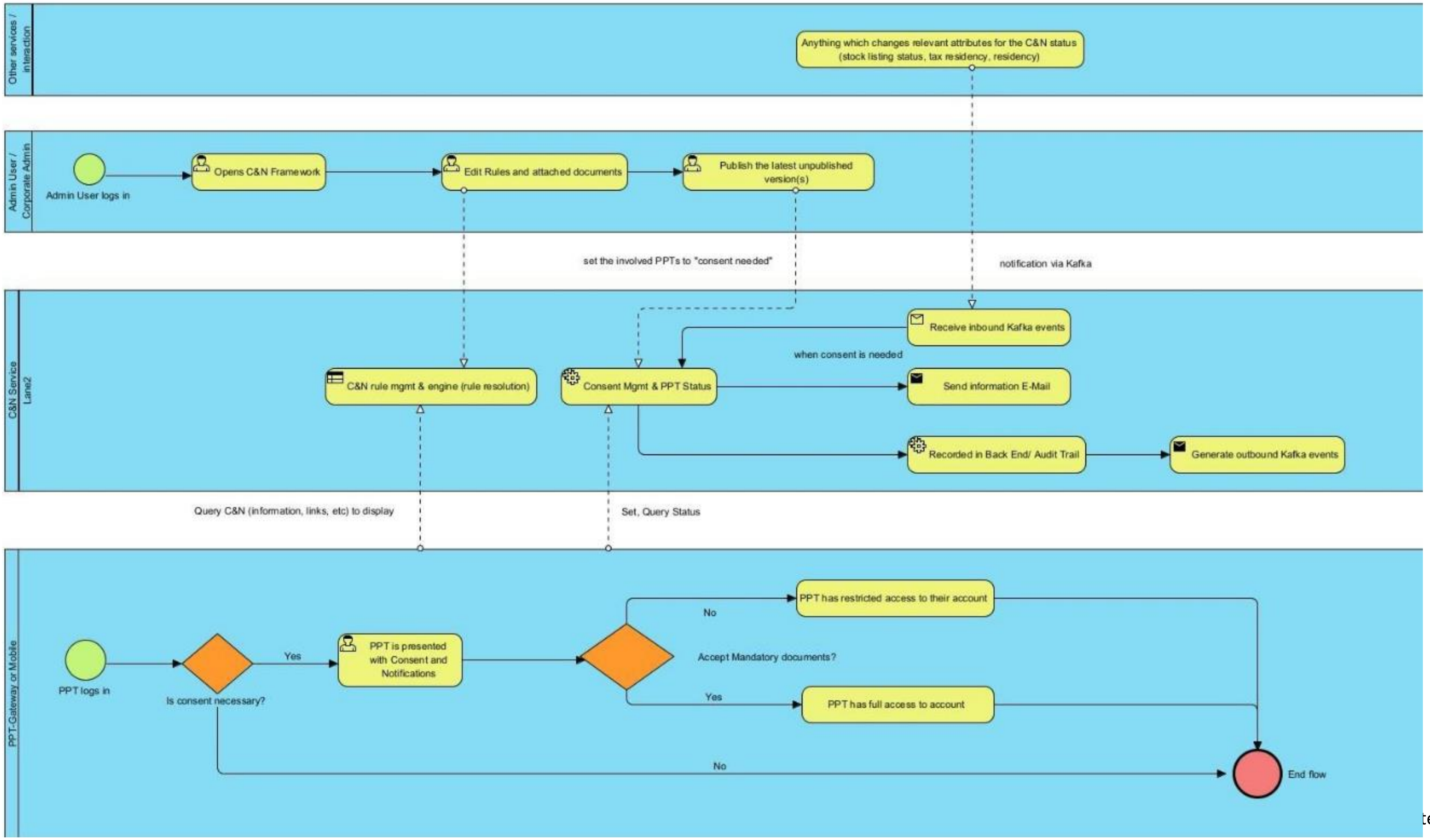
What I want to show you

- Where do you use what, why, how and when
- How to make the most out of use cases and requirements
- Migrate from coder to developer to software engineer

First things first – a kind of intro…

- Real-life is real-life and sometimes things do not work as planned or anticipated. The same is true for requirements & use-cases
- We should try to:
  - use best engineering possible
  - use best tools for the job
  - be as professional as possible
  - design future-oriented not backwards looking software

I love clear specifications


Other services / interaction
Anything which changes relevant attributes for the C&N status (stock listing status, tax residency, residency)
Admin User / Corporate Admin
Admin User logs in
Opens C&N Framework
Edit Rules and attached documents
Publish the latest unpublished version(s)
set the involved PPTs to "consent needed"
notification via Kafka
C&N Service
Lane2
Receive inbound Kafka events
when consent is needed
C&N rule mgmt & engine (rule resolution)
Consent Mgmt & PPT Status
Send information E-Mail
Recorded in Back End/ Audit Trail
Generate outbound Kafka events
Query C&N (information, links, etc) to display
Set, Query Status
PPT-Gateway or Mobile
PPT logs in
Is consent necessary?
Yes
PPT is presented with Consent and Notifications
Accept Mandatory documents?
No
PPT has restricted access to their account
Yes
PPT has full access to account
No
End flow

Reality check

- Designing software (*actually anything where there is interaction between something A and something B*) is not only "design and implementation"

- Designing & Developing SW is more people-skill-centric than most people anticipate

So what are you really doing?

Where do you see yourself now, in 5 years, …

- Coder
- Software developer
- Software designer
- Usability / Experience designer
- Service designer
- Process designer
- Business Analyst
- …

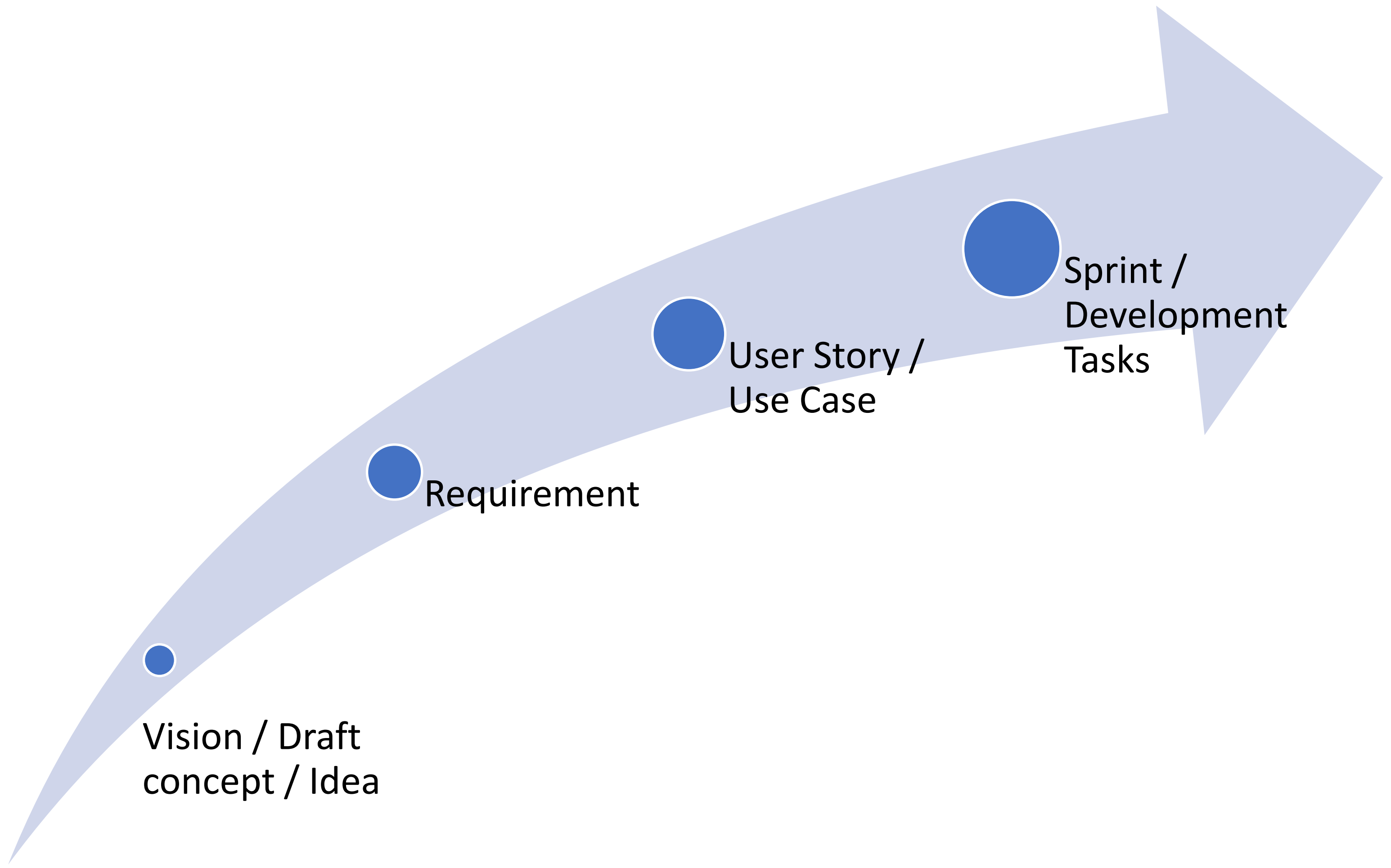
So how is the normal flow when a project starts?

- Someone has an idea
  - Whow – cool! We need a tool / an extension / a solution
    - … (time goes on, initial budget for planning is acquired, solution provider identified, etc.)
    - Requirements are taken down
      - Use cases derived / developed
        - User stories written, duty book, fine-concept, etc.
        - Communication, elaboration
      - …
      - *Change 1*
      - *Change 2*
      - *Legal issue 1*
      - *Legal issue 2*
      - *Budget overrun*
      - *manager changes / has a bad day*

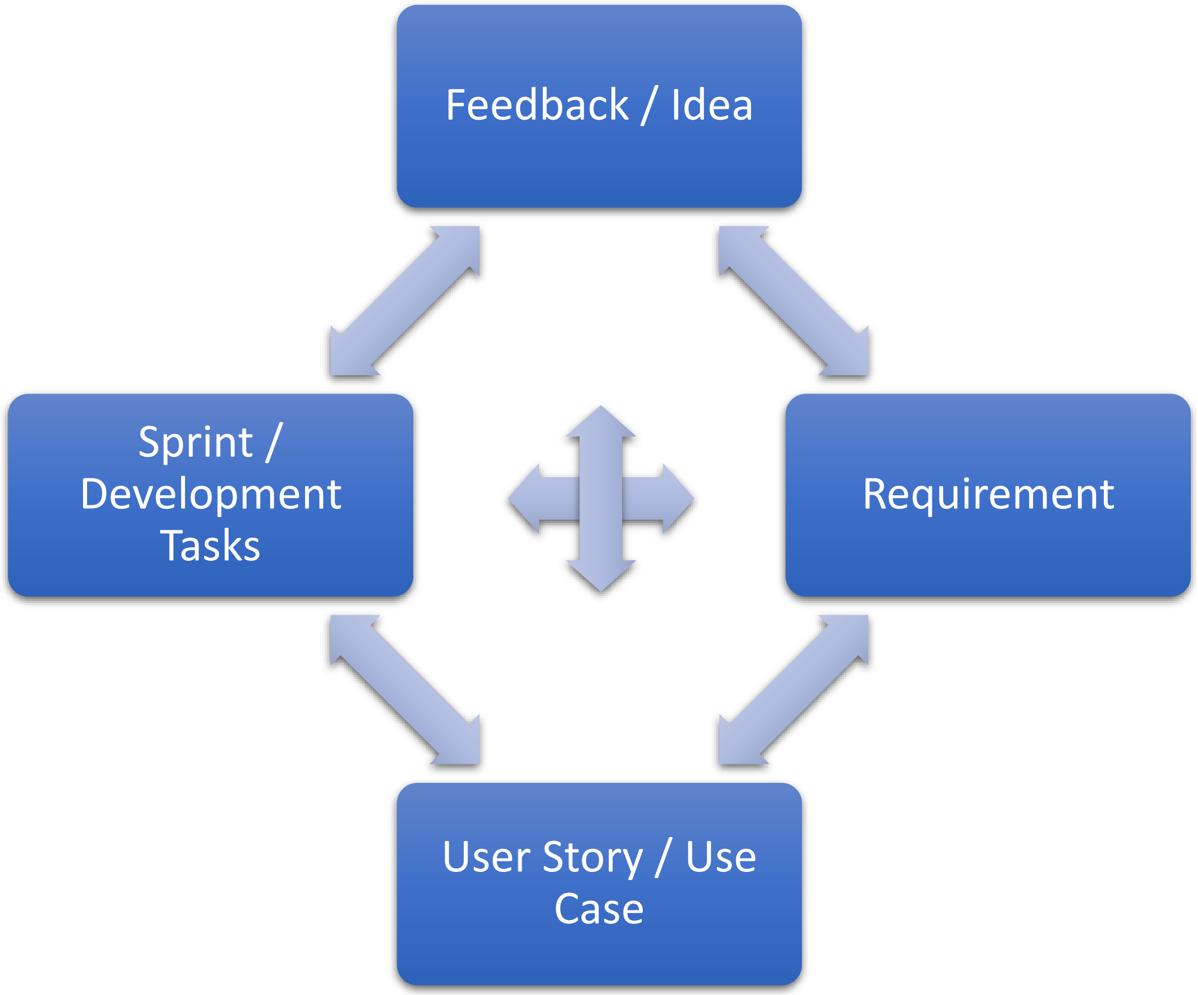
YOU HAVE YOUR GREAT DAY

YOU ARE IN TROUBLE

The ideal world?!?



Our Reality (at least very often)



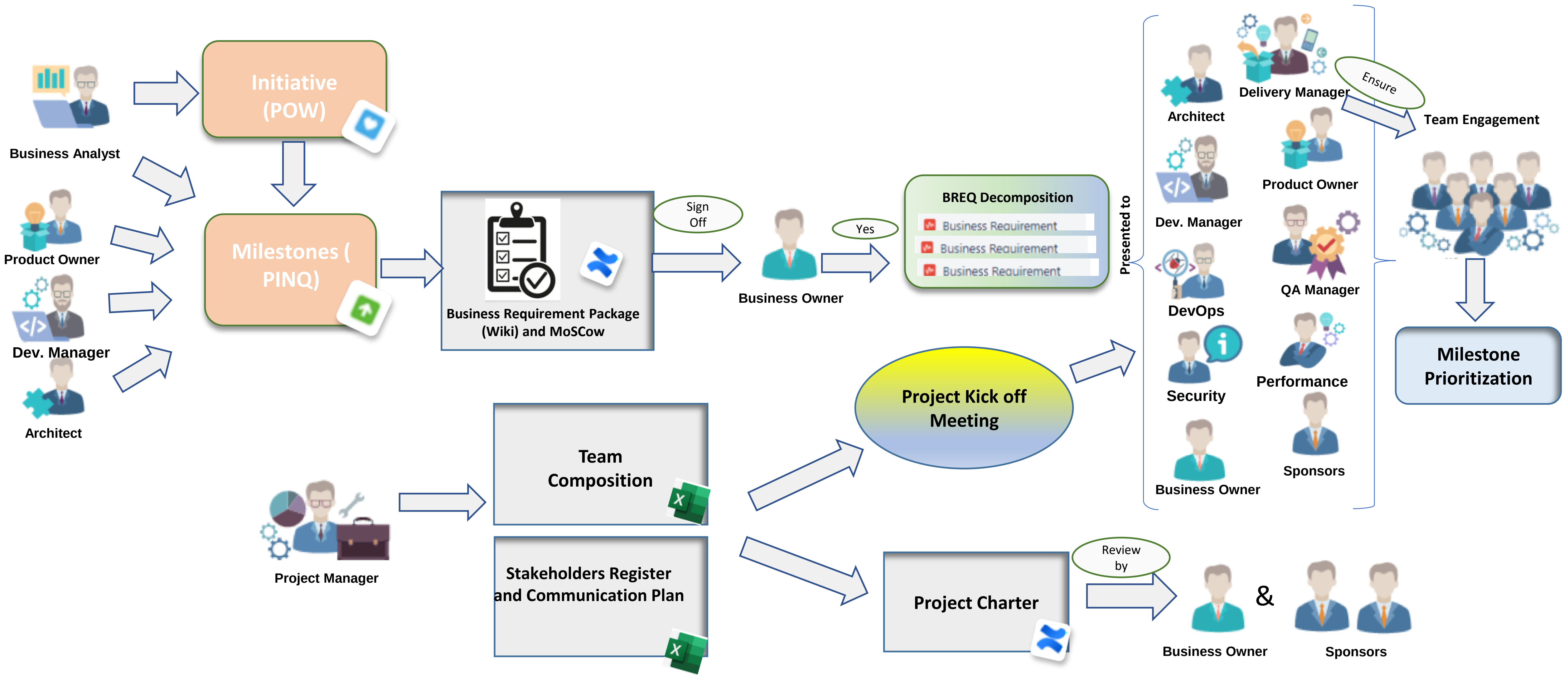
- **Everything influences everything** (dynamic system…)
- **Chaos is somehow normal**
  - The goal is to control (well, better live with) the chaos to a manageable extent

- **Feedback loops are important as this is the decisive moment when we learn**!
  - Design errors during development need fixing (requirements + use cases)
  - Yet they can be beneficial -> lessons learned

- **Every innovation will have setbacks during field-use and will require re-investigation, fixes, etc**.
  - The steam engine took a long time to be as good as it is right now (and now we have to find new ways again…)

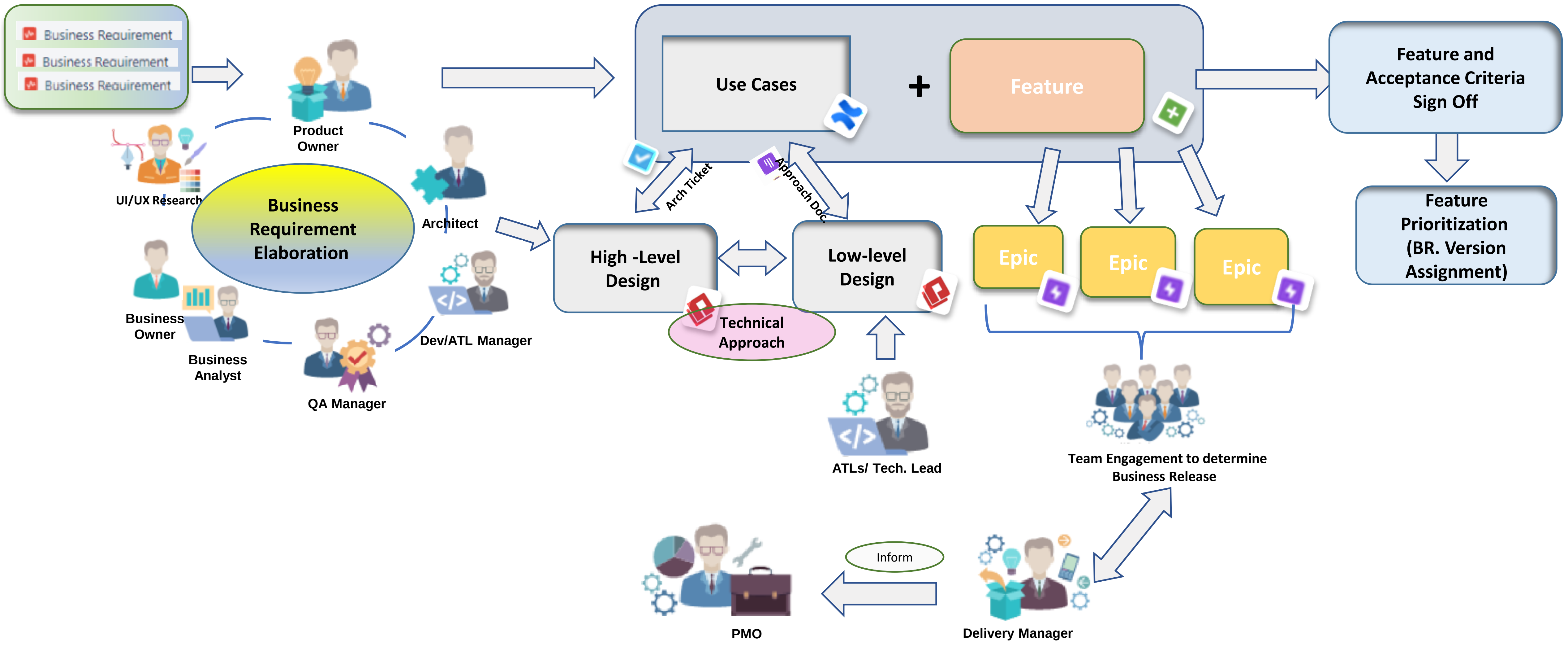
THE UNIVERSITY *of* EDINBURGH informatics

Inception Phase

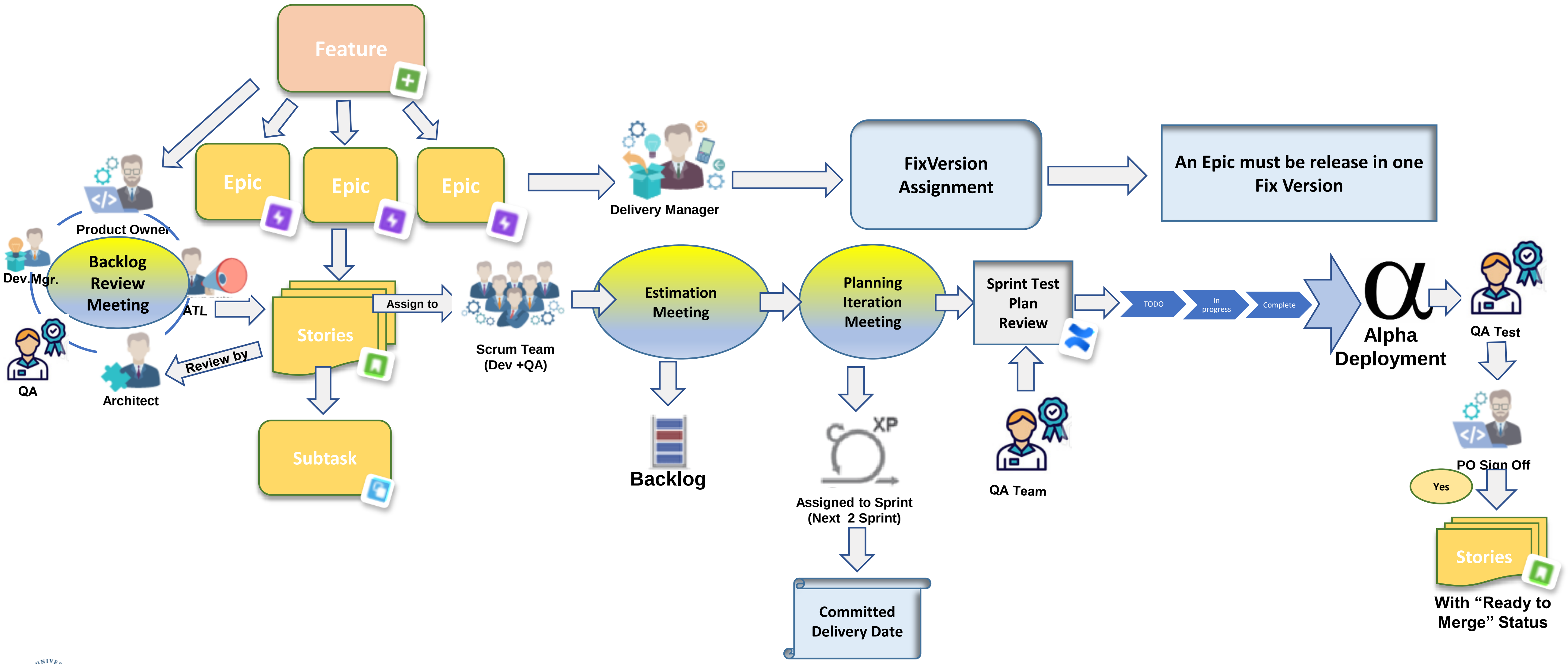


Note: *Only projects that are governed by the PMO will have the Project Charter required. Find more information about this document in the relevant links section.*

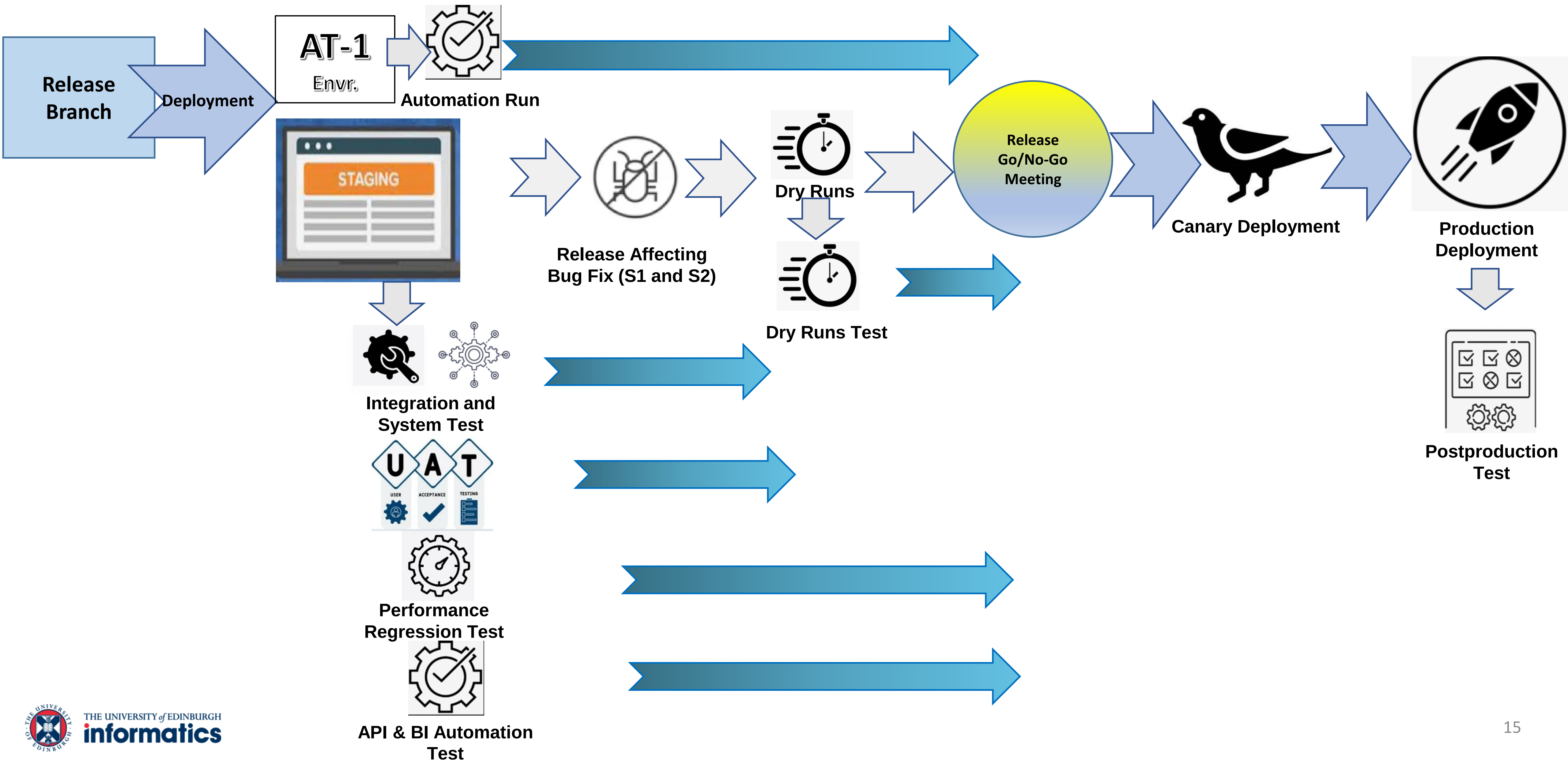
Elaboration Phase



Construction Phase

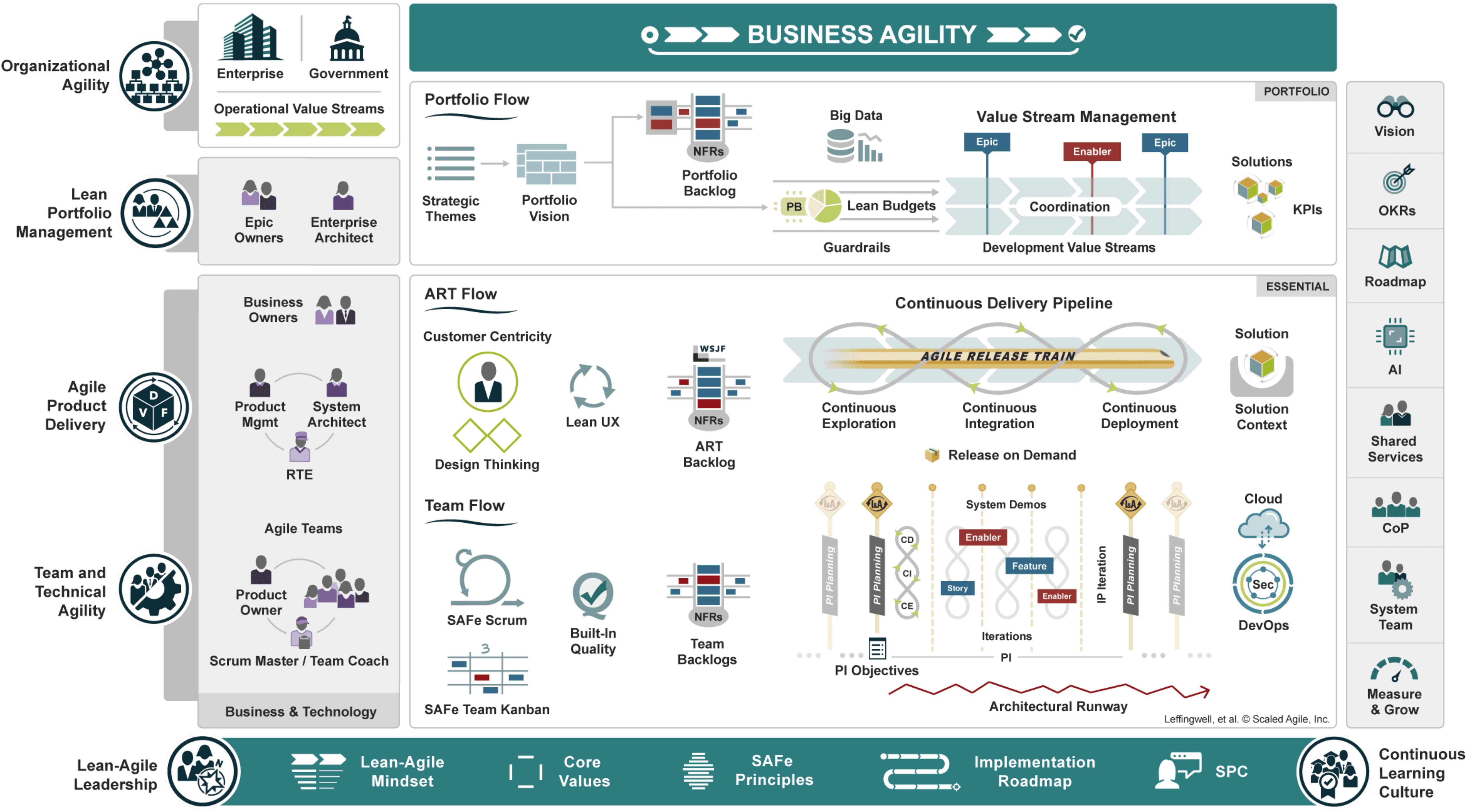


Transition Phase



SAFe (when you do it really large…)

- SAFe 6.0 (scaledagileframework.com)
- Framework for Team of Teams (Tribe…)


BUSINESS AGILITY
Organizational Agility
Enterprise
Government
Operational Value Streams
Lean Portfolio Management
Epic Owners
Enterprise Architect
Agile Product Delivery
Business Owners
Product Mgmt
System Architect
RTE
Team and Technical Agility
Agile Teams
Product Owner
Scrum Master / Team Coach
Business & Technology
Portfolio Flow
PORTFOLIO
Strategic Themes
Portfolio Vision
NFRs
Portfolio Backlog
Big Data
PB
Lean Budgets
Guardrails
Value Stream Management
Epic
Enabler
Epic
Coordination
Development Value Streams
Solutions
KPIs
ART Flow
ESSENTIAL
Customer Centricity
Design Thinking
Lean UX
WSJF
NFRs
ART Backlog
Continuous Delivery Pipeline
AGILE RELEASE TRAIN
Continuous Exploration
Continuous Integration
Continuous Deployment
Solution
Solution Context
Release on Demand
Team Flow
SAFe Scrum
Built-In Quality
NFRs
Team Backlogs
SAFe Team Kanban
I&A
PI Planning
CD
CI
CE
System Demos
Enabler
Feature
Story
Enabler
IP Iteration
Iterations
PI
PI Objectives
Architectural Runway
Cloud
Sec
DevOps
Leffingwell, et al. © Scaled Agile, Inc.
Vision
OKRs
Roadmap
AI
Shared Services
CoP
System Team
Measure & Grow
Lean-Agile Leadership
Lean-Agile Mindset
Core Values
SAFe Principles
Implementation Roadmap
SPC
Continuous Learning Culture

Talking about requirements

- For whom do you do them?
- Do you do them always? When do you do them?
- How / when on the timescale / how often to change

- First things first – people

Talking about requirements

- As the requirements engineer (*very often a role explicitly given away to an external entity or consulting group!*) you are the advocate of the “users” (all levels) towards the implementation / development team
- Consider i.e. fear, job-frustration, anxiety, change-reluctance, … (more on that later)

Some terminology

- A project can be anything where work is done – it doesn't matter which way it is organized or structured.
- In the end it is a "unit of work" which might contain sub-units

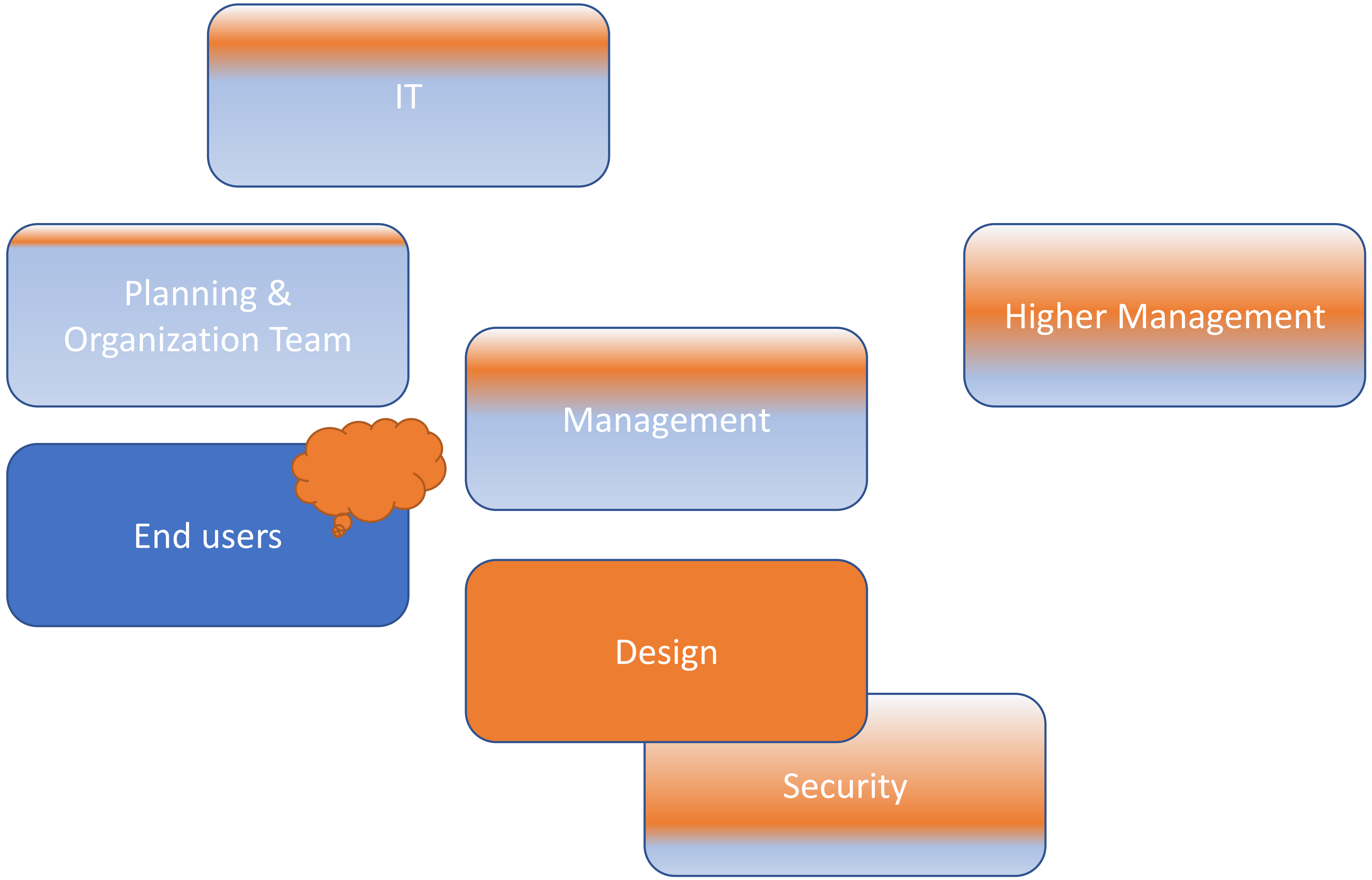
- Every project has a customer which is the one / the instance who has to pay the bills.

Stakeholders in a project

- **Your side**
- team-mates & team-manager
- *It depends…*
  - (Scope / Importance)
    Higher level management
  - Security & policy officer
  - Review management

- **Their side** (the "customer")
- End-users
- Planning & organization team
- Manager
- IT
- Security (data, crypto, consumer legal rights, ethics, etc.)
- Design (UI, etc.)
- Higher management

So whom are you talking to about what?



🟦 Functional Requirements
🟧 Non-Functional Requirements

Live Examples

Requirements – who needs what?

- Users want to find “their” working environment and “their” story
- Planning & organization teams want to see the broader vision (as expressed by the management) and the needed functionality
  - The consider cross-issues as well (sometimes…)

Live Examples

Requirements – who needs what?

- IT needs precise and definitive specs for infrastructure and runtime environments
- Security has to know what happens with data how (person-specific data, customer data, who has access, how long, storage, etc.)
- Management wants to see that you caught the idea and carry it on

You need requirements

- ***You need them – they are your friend!***
- *They are your contract and life-line (at least part of).*
- ***They define** what your software has to fulfill afterwards and ultimately **if you succeed or fail**.*

Live Examples

How?

- Whiteboard, Pinboard with cards or mind mapping (kind of electronic drawing surface)
- Very nice are pads / touch interfaces (you can draw directly)
  - Images are very efficient…
- UML tools are very good for structuring your ideas
  - https://c4model.com/ (C4 model)
- Requirements are often written down in a duty-book / concept, etc. as goals and explicitly requirements your software has to fulfill

What to keep in mind?

- Beware of the poor sods on the other side!
  - no tech-talk or jargon (despite how cool it might be), buzzword-orgies, keep it simple – plain national language
- Try to listen to those who barely speak – they might be the ones with the knowledge and just shy / overheard culturally.
  - Be extra vigilant when managers are present – nobody talks really (cultural thing…).
- People are anxious if they are able to use the new feature / software.

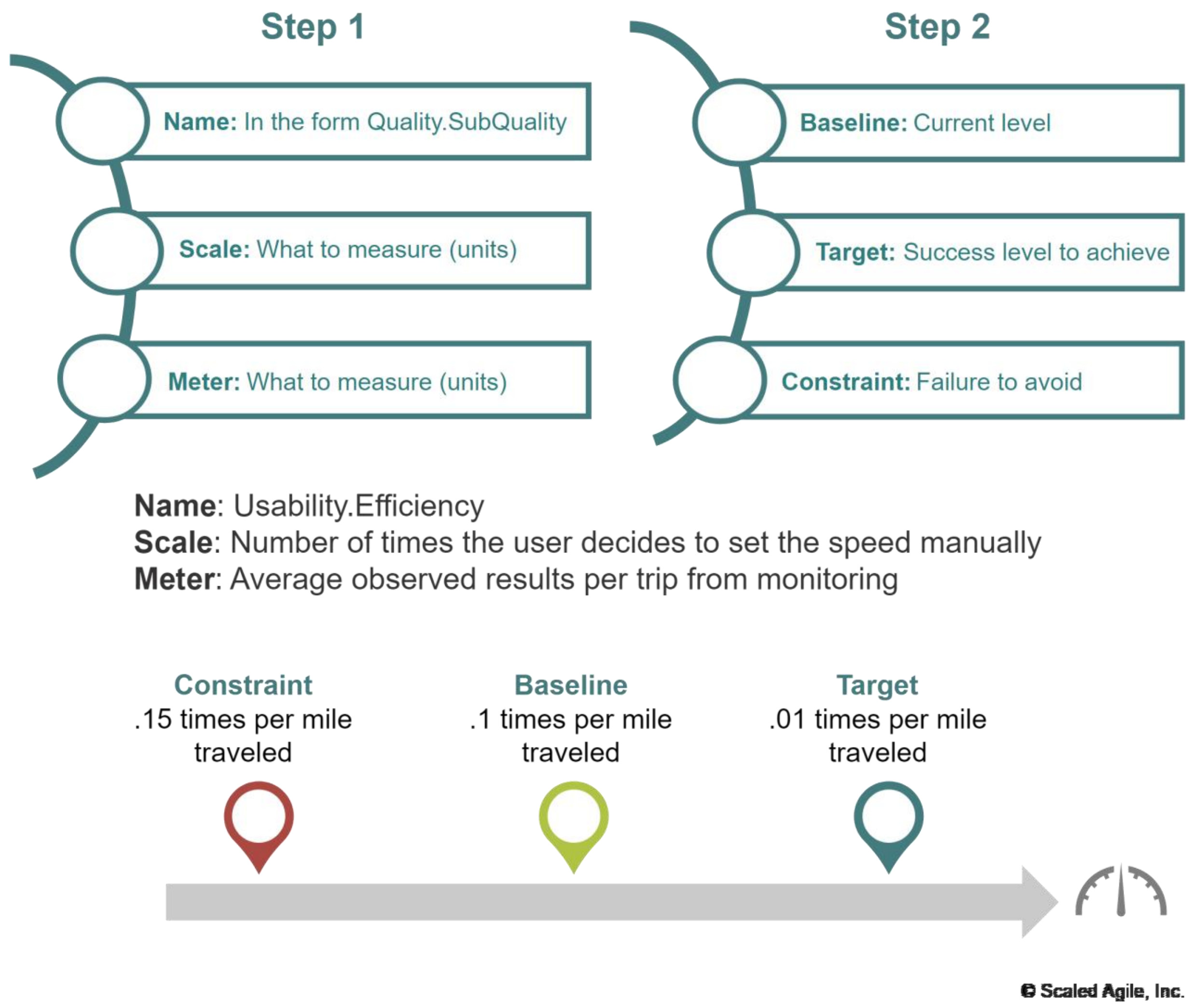
NFRs

- [Nonfunctional Requirements - Scaled Agile Framework]
- Hate them or love them, you need them

- They should be:
  - Bounded (to a context)
  - Independent (of each other)
  - Negotiable (as a crucial aspect of economic performance)
  - Testable (as objective measures)

- Check out the interesting annotation in the next slide

NFR annotation


Step 1
Name: In the form Quality.SubQuality
Scale: What to measure (units)
Meter: What to measure (units)
Step 2
Baseline: Current level
Target: Success level to achieve
Constraint: Failure to avoid

Name: Usability.Efficiency
Scale: Number of times the user decides to set the speed manually
Meter: Average observed results per trip from monitoring

Constraint
.15 times per mile traveled
Baseline
.1 times per mile traveled
Target
.01 times per mile traveled
© Scaled Agile, Inc.

https://scaledagileframework.com/nonfunctional-requirements/#:~:text=NFRs%20are%20persistent%20constraints%20on,Markup%20Language)%20is%20a%20constraint.

Most important

- **Requirements analysis is a perfect moment to take stress and anxiety away and show a way forward which allows growth and especially participation**

-> people design software they are going to use afterwards

- *Different communication levels have different agendas and different requirements for the same thing (e.g. a new feature)*
  - *Management = Cost & Time to market*
  - *Mid-level = functional completeness, will it work with XYZ, can it be extended?*
  - *Users = This is really complicated – how will we be able to use it?*

A simple tip…

- When I jot down notes during a requirements session I have code marks on the list indicating things for later:

⚡	Be careful – problems, issues
?	Needs additional consideration, some more input needed
☺	Makes happy users – give them a dream
♥	I like that very much (personal interest, specific feature, etc.)
✥	General processes are involved and need changing

So, you got your requirements done?

- Congrats – your work just started…
- Now you need the use cases so you can show them and they send you back with homework…
  - That is called “getting feedback”
  - Actually, it is nice – you show people you understood them and give them a vision and they can appreciate and contribute!

Live Examples

What are use cases again?

- They describe things done (actions) by a participant (actor) of the system
- For me they very often describe derived actions as well
    - A booking form is used
        - A specialized booking form with additional security credentials is used
    - A calculation is performed
        - The calculation changes when XYZ as a pre-condition exists
- They show nicely if different users perform the same or similar tasks (which allows for generalization / specialization)

Why I love Use cases

- First of all - I like them and I use them quite often. Mostly during start-up and then later on as a reference (during implementation)

- They help me to group my logical building blocks in software by combining similar functionality

- I can see patterns emerging (e.g. - 4 times a read / write operation, once a read-only -> read-only must be a special case and not the "norm")

Use case benefits

- **If nothing else, use case analysis perfectly clears your mind and structures your thoughts (well, at least it helps).**
- **Use cases help to tackle a complex problem by systematic decomposition into smaller more manageable pieces**

Where do you use them?

- Concept (high-level ones) and sorting out things (usually mid-level)
- Clarification of ideas with customers (more detailed)
- Check that nothing is forgotten

- Mostly (at least for me) they are the pre-stage for more complex structural diagrams

Live Examples

What comes next?

- Full UML diagrams which combine the use cases, the requirements and present flows
  - Activity
  - Composition
  - Component
  - Collaboration
  - Business process

Different project types, different structures

- Depending on the project type requirement analysis and use case usage differs

- Fixed price vs time and material
- Internal vs external projects

- Project vs product development is quite different

- Highly structured (waterfall) vs agile (extreme development)

A word on agile…

- Initial requirement analysis tends to be less then for other project types
- Agile project by their very nature have a tight feedback loop (sprints), so anticipate change
- As more functionality is available earlier for user feedback new requirements are pushed onto the development team earlier as well
- **Despite all flow and dynamic change, a global vision (and global requirement analysis) is still mandatory as otherwise the final solution might deviate (which can be acceptable) too much from the initial goal**