Agile PDF - Foundation Workshops
The Foundation workshops are onsite requirements gathering and planning exercise lasting for a number of weeks depending on project size. The goal is to understand the project as a whole, determine the scope, and estimate the effort for the release regarding must-have and should-have requirements.

The Foundation Phase workshops can be broadly separated into the following activities:

1. Project kick-off
2. Define project governance, organization & structure
3. Define solution goals
4. Define functional requirements
5. Define non-functional requirements
6. Design architecture & system integrations
7. Define environments & tooling
8. Define quality management & testing strategy
9. Release planning & prioritization

Documentation is another important activity that should be done in parallel to the above activities, ideally either as a dedicated time at the end of each workshop or at the end of each day or week. This will be dependent on the workshop agenda, how the project team prefers to work and what fits best for the customer.

These activities do not need to be completed in the above sequence or separately, it’s often more efficient to combine them where appropriate. The most important factor is they are all completed properly so that high quality and accurate deliverables are provided at the end of the Foundation Phase.

Project Kick-Off
The project vision should be presented by the customer (ideally the Project Sponsor) at the beginning of the Foundation workshops and include the history, goals and expected business value to be obtained from the project. To effectively kick-off a project it is important to ensure that all parties and stakeholders understand the shared objectives.

During this workshop, the high-level project plan defined during the preceding Initiation Phase will be presented to the team, as well as the specific goals for the current upcoming release. This sets the context for the subsequent workshops.

Define Project Governance, Organization & Structure
The project governance, structure, and organization should be defined during the Foundation workshops incorporating all project stakeholders. Roles and responsibilities should be documented and clearly communicated, for example, responsibility for frontend code, cross-browser compatibility testing, performance testing, the definition of acceptance criteria, and infrastructure. This is documented as a RACI matrix in the Project Management Definition deliverable.

Regular project meetings and communication channels should be defined and agreed. For example:

- Instant messaging/voice communication tools (e.g. Skype)
- Regular working group calls, e.g. weekly status call for leads of each work stream
- Regular customer reviews at the end of each sprint
- Regular sprint planning/retrospective sessions
- Steering review meetings

**Define Solution Goals**

Understanding the goals of the new solution is very important as these should be used as a reference point for decision-making, e.g. during requirements prioritization. Therefore it's important these are understood upfront during the Foundation Phase workshops, so everyone is aligned. It's equally important to identify if goals are unrealistic and mitigate these expectations as early as possible, resetting goals appropriately.

Typically solution goals are centered around:

- Pain points
- Revenue / key performance indicators (KPIs)
- Competitive advantage

Additionally, it's equally important to understand the consequences of NOT doing the project and building the new solution. Often these are very powerful drivers in themselves, and it's very important to understand these, so it's recommended these are also explored during this workshop.

**Pain Points**

If there's an existing eCommerce solution these often include:

- Performance/scalability issues or bottlenecks
- Stability/maintainability issues
- Flexibility/extensibility issues
- Integration issues
- Time-consuming or error-prone manual processes that could be automated
- Things they would like to remove/do differently
- Things they wish it did
- User interface/site design issues

Even if there's no existing solution, there will likely be pain points in the customer's business that the new solution may be able to resolve, perhaps that the customer hasn't considered and may be identified as they explore the new solution in more depth.

**Revenue / KPIs**

Revenue targets and KPIs typically go hand in hand with goals and objectives, and may even be outlined as part of the success criteria. These metrics can also be helpful to determine the size and scale of the solution, along with projected growth. Typically these are marketing, sales, or traffic related.
Key metrics to establish if possible include:

- Daily/weekly/monthly/yearly sales
  - Gross and net
  - By volume and amount
- Average order value (AOV)
- Average order size (number of items)
- Margins ($ and %)
- Conversion rate
- Abandonment rate
- Bounce rate
- Site visits
  - Simultaneous
  - Daily
  - Weekly
- Traffic source
  - Native
  - Search
  - Paid-search
  - Pay-per-click
  - Email/promotion
  - Referral
- Time-on-site and number of page visited
- New vs. returning customers
- Stock/inventory levels

**Competitive Advantage**

Understanding the customer’s competitors and their strategy to compete against them is very useful. This can be done by:

1. Outlining all known and up-and-coming competitors and ranking them appropriately, e.g. regarding their eCommerce strengths/weaknesses.
2. Explore how they compete with each competitor, and whether there are areas, they are trying to catch them in, as well if there are areas they are trying to be an innovator or gain a competitive advantage.
3. Ask the simple question: what is the best thing the customer’s biggest competitor is doing that they aren't?

A thorough understanding of the customer’s competitive situation can help the project team guide the new solution to maximum advantage, e.g. if a competitor is particularly strong or weak in an aspect of their eCommerce offering, these can be used as reference points during subsequent requirements analysis workshops.

**Define Functional Requirements**

During the Foundation workshops, the customer’s business and technical representatives further elaborate upon the features for the current release as defined during the preceding Initiation phase. These sessions will be scheduled as defined by the workshop agenda that was agreed before the workshops commence.

This guide will assume paper cards are being used to capture requirements during the Foundation phase, although other methods can also be used.

**Feature Presentations**
The following presentations are required to communicate the business and technical requirements of a feature:

**Business presentation:**

The customer representative responsible for the respective feature should communicate all business requirements. Examples may include:

- Feature overview
- Expected business value
- Mock-ups (screenshots) or layouts

**Technical presentation:**

The customer representative and/or the supporting technical authority should communicate any known technical considerations relating to the requested feature. Examples may include:

- Rules/restrictions
- Entity relationship diagrams
- Data flow diagrams
- System diagrams
- Specific non-functional requirements
- Analytics specification
- Technical dependencies
- Technical value to help subsequent prioritization

The business and technical presentations are often combined to make sessions more efficient.

**Q&A Session**

Following each presentation, a question-and-answer (Q&A) session allows the project team to immediately clarify requirements and possibly suggest alternative options to ensure the customer’s requirements are fully understood, and the best solution is decided. It’s important to do this while the information is fresh in people’s minds and the relevant people are available. This is also a good opportunity for the project team’s Business Analyst to clarify the information captured on the paper cards during the presentations.

**Requirements Documentation**

Once the project team can understand the requirements to an appropriate depth, then the requirements can be documented. Further elaboration of each requirement (e.g. acceptance criteria) will be defined later during the Engineering phase as needed. This further level of detail is not required at this stage. The project team only needs to understand the requirements to enough depth to be able to estimate effort/complexity, understand value, and any dependencies.

During the business and technical presentations the project team’s Business Analyst should decompose features into sub-features and document each on paper cards. A card should also be created displaying the associated feature title for use on the release wall (if not prepared in advance).

If a feature is deemed to be too large or complex, then the project team may split the feature into separate features. This can only be achieved if each feature, once split, still provides value and can be delivered independently.

**INVEST**

The acronym INVEST can be used as a checklist to assess the quality of a feature, sub-feature or user story. This suggests a well-defined feature/subfeature/user story should be:

"I" ndependent (of all others)
“N” egotiable (not a specific contract for features)
“V” aluable (or vertical)
“E” stimable (to a good approximation)
“S” mall (so as to fit within a sprint)
“T” estable (in principle, even if there isn’t a test for it yet)


This should be used as a guide. However, it’s not always possible to follow exactly, e.g. sometimes sub-features are dependent on another sub-feature being implemented first.

Here is an example sub-feature card showing how paper cards can be used to document requirements:

Front side:

ID-123
Feature/requirements tracking tool ID (if available)

Persistent cart
Sub-feature title
Cart & Checkout
Feature title

Reverse-side:

- Cart should be restored when logging in from different browsers/devices
- When logging in, items from actual cart should be added to restored cart
- Remove old cart

User stories and technical tasks

The following example illustrates the relationship between a feature and a sub-feature in paper card form:
Product Catalogue

*Feature*

Product data import

*Sub-feature*

Manage target groups

*Sub-feature*

Phased out products

*Sub-feature*

Etc.

*Sub-feature*
Dependencies between sub-features can be visualized through the use of stickers on the cards that display appropriate comments. It is important that dependencies are documented to avoid features being dependent on other features being selected during release planning.

Estimation

Once each feature has been decomposed into sub-features they can be estimated. If possible, ideally the estimations are done in view of the customer, so they can immediately answer questions coming up during the estimation discussions. This not only ensures that all information is fresh in the minds of the project team as they are estimated but also that each feature is fully documented before estimation while displaying transparency during the process. Sufficient time should be allocated for estimation activities to occur following each feature presentation.

**Estimation Approaches**

There are different approaches to estimation, and these can vary depending on the project team’s preferences. Here we present a recommended approach that has been successfully used in many projects (commonly referred to as ‘planning poker’):

1. With all people in view of the sub-features to be estimated, a designated Scrum Master facilitates the estimation activities.
2. The Scrum Master or Business Analyst should provide an overview of the sub-feature.
3. The Scrum Master decides whether the team is ready to estimate:
   a. Ensures that all attendees are comfortable with providing estimates. If any attendees are not comfortable with providing an estimate, then further discussion should occur.
   b. If further clarification or feedback is still required and the customer is unable to provide enough information to enable the project team to estimate then the feature paper card (and hence all subsequent related sub-feature cards), should be marked as blocked until further information can be provided. Once sufficient information has been provided then the relevant sub-feature/feature should be revisited either during a separate consolidation workshop or ad-hoc as required.
4. During the estimation activity the project team focuses on a single sub-feature and estimates using planning poker:
   a. Provide a five-second countdown to select estimates
   b. All people display their estimates at the end of the five-second countdown
   c. Read each estimate aloud while assessing the range of estimates from the lowest to highest
   d. Request feedback from those who have provided the lowest estimate
   e. Request feedback from those who have provided the highest estimate
   f. Refine estimates as a group and reach a consensus
   g. Document the agreed final estimate on the associated paper card

Once the sub-features have been estimated, the paper cards should be placed on the release wall within the ‘To be scheduled’ area underneath the associated feature title card.

Define Non-Functional Requirements

Non-functional requirements (NFRs) are equally important (if not more important) than functional requirements. You can have the most perfect functional solution in the world, but if it isn’t accessible because it can’t handle the number of users, not only does it not work, but it can also have serious impacts on brand perception, lost customers and ultimately revenue.

Therefore the project team needs to ensure non-functional requirements are captured during the technical presentations. Typically this is the role of the project team’s Architect to ensure these are captured. Additionally, there
should be dedicated workshop sessions specifically to discuss the customer’s non-functional requirements for the solution. These usually focus on what is commonly called the ‘ilities,’ e.g. scalability, maintainability, operability, upgradability and so on. It’s important these are covered separately as NFRs tend to overarch all individual functional requirements.

NFRs are a vital input into the architecture and infrastructure workshops, and they can also have a significant impact on planning and costs. Additionally, NFRs need to be known before development can start (so before the Engineering phase), as they have a significant influence on architecture, technical design decisions, and testing effort. So it’s essential these are thoroughly captured during the Foundation phase and planned in from the start of the project.

The project team’s Business Analyst and Architect should document non-functional requirements on paper cards in the same manner as sub-features so that they can be prioritized and planned during the Foundation workshops. Non-functional requirements may also be documented within acceptance criteria of a feature/subfeature to ensure that they are met if essential for implementation. The information collected should form the basis of the Non-Functional Requirements document deliverable to be provided by the project team’s Architect.

Some considerations for the definition of non-functional requirements:

- Performance
- Scalability
- Capacity
- Availability
- Maintainability
- Security
- Regulatory

Rather than try to estimate each NFR individually which can be challenging, a better approach is often to create a buffer to incorporate groups of NFRs together, e.g. a Story Point buffer. For example, a 100 Story Point buffer for performance can be assigned to the release, then used up as the Engineering phase progresses through the sprints depending how much effort is spent on performance in each sprint. This would then also count towards the completed Story Points in each sprint and therefore also the sprint velocity and release goal.

**Design Architecture & System Integrations**

High-level physical and logical architectures and integrations with third-party systems should be investigated during the business and technical presentations. Early design activities such as core data modeling activities can also be undertaken where it adds value to the workshops, e.g. facilitates discussions or explains concepts. For example, this is commonly done for catalogs, products, content, pricing, etc.

The project team should pay particular attention to any required system integrations and ensure that the impact of any such integrations is sufficiently understood. If the project involves integrating with a third-party system which is not managed by the customer or project team, then representatives from the third-party should be in attendance during the relevant feature business and technical presentations to provide additional feedback if required.

**Define Environments & Tooling**

Infrastructure environments and tooling are essential to the project. For example, if infrastructure environments are not ready on time they will have a major impact on overall project timings, e.g. if there are no test environments to deploy to then no testing can occur, and development is held up. There is always a lead-time to setting up these environments, so they need to be specified as soon as possible, and timings built into the project plan.
It is recommended that the project team works closely with the customer to define required infrastructure and environments (e.g. development, test, pre-production, production). The requirements for these environments are likely to be dictated by the NFRs and prioritization of features.

Tooling is also a very important topic that can easily cause delays and frustration if not properly considered and planned up-front. For example, the project management tools (e.g. Jira), the development tools (e.g. IDE, continuous integration, source code repository), testing tools (e.g. performance testing), etc. There is always effort to set these up at the start of a project. Changing them mid-way through a project is very time consuming and will likely cause significant delays.

**Define Quality Management & Testing Strategy**

It’s very important to define and agree how the solution will be tested throughout the project to ensure the quality goals are met. Ultimately the customer needs to decide the quality goals as it’s an investment that requires effort and therefore budget. However, it’s strongly recommended this is properly considered and planned. Quality needs to be baked-in from day one; it’s much more expensive to try and fix quality later.

Specific workshops should be allocated specifically to cover the different types of testing and ensure responsibilities are agreed. A RACI matrix in the Quality Assurance Definition is a good method to document who is responsible, who approves, who is consulted and who is informed.

Testing topics include:

- Feature tests
- Integration tests
- Acceptance tests
- Deployment tests
- Performance tests
- Cross-browser compatibility tests
- Post-sprint acceptance tests
- System integration tests
- Authorization tests
- User acceptance tests
- Security tests

Test data should also be discussed and planned during the workshops. Preparing test data can take a significant amount of effort. The quality of the data is also very important, and data cleansing can also be very time-consuming. This can be a major source of delays and quality issues later if not planned properly.

Infrastructure environments are also a key input here. It’s essential that appropriate testing environments are available when required. Tooling is also a key consideration and needs to be discussed/agreed with the customer. The defect management process should also be agreed with the customer.

This information forms the basis for the Quality Assurance Definition (QAD) deliverable, to be published by the project team’s senior testing representative. The QAD template contains additional information about these testing topics.

System integration testing and integration testing test how the solution interacts with other systems, but integration testing is performed by the project team and may use mocked integration so that it is not impeded by external systems. Whereas system integration testing is performed by the customer and also includes testing of interactions in the external systems that cannot be simulated through mocks.
Release Planning & Prioritization

Once features/sub-features are present on the ‘To be scheduled’ area of the release wall the project team, and customer can collaboratively begin the prioritization activity. This can be an on-going activity or may be undertaken once all features/sub-features have been estimated. It is preferred that this is an on-going activity to ensure the customer is aware of the running total of the estimates produced to compare this figure against the targeted release capacity as the Foundation workshops progress.

The project team and customer should prioritize the features within the ‘To be scheduled’ area of the release wall, moving the associated paper cards to reflect the designated priority of each feature while considering all previously highlighted dependencies.

With the capacity for each release displayed on the release wall with the must-have/should-have subtotal split, and estimates documented for each feature/subfeature, and the ‘To be scheduled’ cards prioritized, the customer can designate sub-features as must-have or should-have and begin to move the cards on the wall to create the release roadmap, keeping within the capacity and agreed must-have/should-have ratio proportions (ideally 60/40 ratio %).

When managing the must-have/should-have ratio the dependencies of each sub-feature should be considered to ensure that a must-have is not dependent on a should-have as this would result in a blocked must-have sub-feature.

It’s recommended that a single person is responsible for moving cards to prevent confusion and sub-features being mishandled during the release planning activity. Mishandling feature cards can result in requirements being lost which can have a significant impact.

Once the targeted release capacity is reached with the appropriate must/should-have ratio, the release roadmap is now planned. The targets should be met exactly or slightly under, never exceeded.

With this release roadmap, the customer and project team now agree on the initial plan for what will be developed for each release, the prioritization within each release and the agreed contingency within each release (the should-haves). Of course, this is only a plan, and the content of each release is not fixed, and the capacity target is also variable as it’s only been estimated and is subject to a wide variety of factors, hence the need for contingency. The customer can also define new requirements and change prioritization during the release with the agreement of the project team. So the release roadmap constantly evolves as the project continues.

Each sub-feature card should then be marked with the planned release number and whether it’s a must-have or should-have.

Finally, the paper cards should resemble the following:

**Sub-feature card**

Front of card:

<table>
<thead>
<tr>
<th>ID: 123</th>
<th>Feature/requirements tracking tool ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Release</td>
</tr>
<tr>
<td>Persistent cart</td>
<td>Sub-feature name</td>
</tr>
<tr>
<td>Dependency</td>
<td>Cart &amp; Checkout</td>
</tr>
<tr>
<td>13</td>
<td>Estimate</td>
</tr>
</tbody>
</table>

Reverse of card:
The following diagram displays an enlarged view of the completed release roadmap with features preselected for an example Release 1, split into ‘must have’ and ‘should have’ along with features that have been moved to a future release.

Once the release roadmap has been agreed and published, it should be distributed to all members of the project team for further review before final confirmation with the customer as final deliverable of the Foundation phase. This is a critical input into the Engineering phase, so it’s important to check thoroughly and get buy-in from the whole project team.

**Recommended Practices**

- Agile PDF - User Story
- Product Catalog
- Media Management
- Non Functional Requirements
- Classification System