|  |  |
| --- | --- |
| Arrow: Counterclockwise curveArrow: Slight curve  **pre** |  |
| Arrow: Slight curve  **post** |  |

|  |
| --- |
| ****Planning Material**** **Lightweight**  **Disposable** |
|  |
| ****Implementation In-progress Material**** **Lightweight**  **Disposable** |

|  |
| --- |
| ****“Done” Material**** **Version-controlled with the Increment**  **User-centric/Audience-specific** |
|  |
| ****Hand-off Material**** **Version-controlled with the Increment**  **User-centric/Audience-specific** |

****The Art of Agile Product Documentation****

A Sense-making workshop to simplify *valuable* documentation & reduce *wasteful* documentation

## How to Use

1. Check here that you have all the materials you need: [https://davesabine.com/The-Art-of-Agile-Product-Documentation/](http://bit.ly/2qUPPAO)
2. Conduct the workshop with your own teams & stakeholders.
3. Tell me how it goes via [Twitter](https://twitter.com/DaveSabine) or [LinkedIn](https://ca.linkedin.com/in/davidsabine).
4. Ask questions, report a typo or problem, or suggest changes here: [Submit New Issue](https://github.com/DavidSabine/The-Art-of-Agile-Product-Documentation/issues/new)

## Learning Objectives

1. Reframe our understanding of documentation with respect to knowledge work in complex environments. (Documents are not truth. They are snapshots of current understanding. If not treated carefully, they create fiction not transparency.)
2. Reframe our understanding of 'implementation' — when does it occur in product development? (Hints: it isn't a phase or project milestone; it is every moment in which a decision is codified in \*the product\*.)
3. Compare the purpose of artifacts/documents produce pre and post implementation. (Documents pre-implementation do not represent decisions; they represent, at best, incomplete information. Documents create at or after the point of implementation are obsolete the moment they are produced.)
4. Appraise commonly-used documents with respect to customer-value — in contrast to perceived business/process 'importance'. (Documents are often produced because someone demanded they be done; but many documents are not the artifacts that any customer is willing to pay for. How might we focus on documentation which has actual value?)
5. Consider and describe ways each artifact may be or simplified. (Like eliminating a Business Requirements Document in favour of a flexible/dynamic Product Backlog, how might an Agile team simplify the design and production of necessary artifacts/documents?)

## Segment 1: Connection/Reflection (~8 minutes)

1. In segment 1, everyone in the Product Development organization will make visible all the documents currently in use.

**SCREEN COMPOSITES**

**USER STORY**

1. If conducting this activity with larger groups, subdivide into smaller groups approximately 5±2 and provide each team/table-group with the blank worksheet from page 1 of this document.
2. Suggest they use small sticky-notes with one idea/document per note.
3. Encourage the participants to each take a few stickies and write quietly at first. (This will speed up the activity.) After the most obvious items are noted, discussion will naturally occur as the groups think more deeply and uncover less obvious types of documents.
4. Place the stickies anywhere on the canvas. (Ignore the quadrants for now, the quadrants become meaningful in next segments.)
5. Ask participants to list all documents which come to mind – it is not helpful to filter, discuss, or debate in this segment. Some examples might be helpful such as “BRD” or “Project Plan”, “UI design”, “test plan”, “tech spec”, “SoW”.
6. Suggest they be specific. And general. Suggest they list internal documents, and client-facing documents, and between-team documents, and short-lived documents and long-lived archives, etc.
7. Some less obvious documents can be suggested: email, chat messages, contracts, flow diagrams, time sheets, audit evidence, compliance measures, test results, experiments, prototypes, source code, Master Service Agreements, payment receipts.
8. Groups can often identify many dozens in just a few minutes.

Before moving to segment 2, ensure each group has at least these on their worksheet:

* Contract
* Requirement
* Artboard
* Test plan
* Test result
* Source code
* Usage guide
* Installation guide

## Segment 2: Concept (~10±5 minutes)

The objective of segment 2 is to clarify the ‘Point of Implementation’. When the group has a shared understanding of that phrase, they will more easily identify the types of documents produced before, during, and at-or-after implementation.

Clarity of this concept is important for successful facilitation of the exercise, please forgive this lengthy explanation:

The ‘Point of Implementation’ is the point in time that a new behaviour, feature, requirement, element is codified into the product. If thinking in terms of a new feature, the Point of Implementation occurs when the source code is being altered to produce the new behaviour – the new feature is said to have been “implemented” when the Product Developer or user can execute the new feature. If thinking in terms of a new user interface element, the Point of Implementation occurs when the user interface is being altered to display the new design. If thinking in terms of a bug fix, the Point of Implementation occurs when the source code is being adjusted to correct the undesirable behaviour.

Product Development is always, actually, a series of adjustments – we can say, “a series of implementations” – as each new behaviour is put into effect. By analogy, it would be odd to say that “I will implement this jigsaw puzzle”; but it would be quite normal to say, “I will implement a piece of the puzzle”. Likewise, it would be odd to say, “I will implement this new Product”; but it would be quite normal to say, “I will implement a new feature of the Product”.

In this way, each Point of Implementation for an adjustment to the Product follows some activity, such as discussion, designing, experimentation, and testing. And each Point of Implementation for an adjustment is followed by automated testing, deployment, and real-world use of that new feature.

In practical terms, a Point of Implementation occurs every time a Product Developer commits new lines of source code into their code repository. Even if that commitment includes a single line of code, that is the ‘Point of Implementation’ with respect to a feature, or requirement, or bug fix, or adjustment.

Segment 2 ends when everyone in the room understands this definition of ‘Point of Implementation’ – even if they do not yet agree or subscribe to it, they can understand and use this definition for the remainder of this workshop.

In work environments where sequential/phased development patterns have been practiced (i.e. waterfall) talking about the Point of Implementation as described above can cause significant cognitive dissonance and argument. The reason is that ‘implementation’ is often a label given to a period of time (a ‘phase’ of a project) in which a very large batch of source code is produced.

Likewise, the word ‘execution’ is also misused for similar reasons. For example, every Software Engineer knows that their Product can be executed thousands of times per day, yet Project Planners tend to think of ‘execution’ as a long period of time, usually many weeks or months, in which a very large batch of requirements are codified.

## Segment 3: Affinity Mapping (~4 minutes)

1. Draw attention to the labels, ‘pre’ and ‘post’, on the canvas.
2. Suggest the participants move and group their sticky-notes accordingly. All documents which are done prior to a Point of Implementation shall be moved above the horizontal line; all documents which are done at or after a Point of Implementation shall be moved below.

Examples:

A requirements document (like a BRD, Product Backlog Item, User Story, architectural design guide) is most-often used as a planning document before implementation.  
[Above the horizontal.]

A project plan is, by definition, done before any implementation occurs.  
[Above the horizontal.]

Source code is often contributed at the point of implementation.  
[Below the horizontal.]

A ‘change log’ is often produced after new source code has been implemented. [Below the horizontal.]

An automated acceptance test (TDD or BDD) is often contributed at the point of implementation.  
[Below the horizontal.]

As in the previous segment, cognitive dissonance and argument may occur – usually in work environments where the release of source code (into production environments) is considered “implementation”. In waterfall settings, the point in time when source code is committed (i.e. implemented) and the point in time when that same source code is launched (i.e. released) may be segregated by days or months. In agile environments with healthy DevOps practices, those points in time (between implementation and release) may be minutes or hours.

Nonetheless, the Point of Implementation occurs, not when the code is released into production environments, but when the code is committed to the repository – that is when the new functionality is said to be “implemented in the Product”.

Segment 3 ends when each group has finished moving their stickies and when the facilitator confirms that at least these are above the horizontal:

* Contract
* Requirement
* Artboard
* Test plan

And at least these are below:

* Test result
* Source code
* Usage guide
* Installation guide

Some may argue that “Test plan” belongs below the horizontal. If so, request their patience on the matter and make note to cover “test-first” or “test-driven” development in a subsequent discussion.

## Segment 4: Concept (~10±5 minutes)

In Agile Development environments, we actively seek and practice ways to maximize simplicity and value while also keeping waste to a minimum. The following guiding principles will help a team to develop those agile practices as they seek to maximize simplicity and value of their work and product.

1. Documents which inform decisions leading up to a Point of Implementation are kept:
   * Lightweight
   * Disposable
2. Documents which record the condition of the Product \*at\* a Point of Implementation are kept:
   * Version-controlled with the Increment
   * User-centric/audience-specific

Until the Point of Implementation of a feature/adjustment/bug-fix/requirement, all dialogue about said feature is conjecture. (Speculation, not specification!) Documents may be *informative* (at best) but do not represent *decision*. To think, in a complex problem realm such as Product Development, that pre-implementation documents represent *decision* is risky and incompatible with reality. Pre-implementation documents represent our best hypotheses (guesses) about what our users might need. Until we codify (make real) our best understanding of their needs into the Product (i.e. the Point of Implementation) it is not possible to test those hypotheses with actual users.

We embrace the reality that our user feedback will reveal more truth and cause important changes in our understanding of their needs and the ways they use our product. In other words, much of the detail we might write into pre-implementation documents *will change* when we implement and then test in the hands of real users. So, in order to keep waste to a minimum we keep pre-implementation documents lightweight and disposable. Lightweight, in the sense that we are content with a pencil sketch of User Interface and resist the urge to make a high-definition/photorealistic rendering. Disposable in the sense that we embrace the notion that we are probably wrong (at least partially) and we’re ready to throw out our idea if a better idea may later arise.

When we implement, a decision is codified — all information and dialogue to that point in time is captured and realized in the implementation. The implementation is therefore a record of a set of decisions. We want the ability to mark said decision-set as distinct from other sets — both in time and proximity. We also want the ability to undo the implementation with ease. Thus, version-controlling each increment of Product enables us to implement our decisions in small, reversible steps.

As well, we remember our highest priority is to satisfy the customer through early and continuous delivery of valuable increments — generally speaking. Therefore, each version-controlled increment, and all of its related post-implementation documents, must satisfy their users.

* Who is the code for? (Developers) So make the code developer-centric and meaningful to that audience.
* Who is the UI for? (Users) So make it excellent for those people.
* Who is the audit evidence for? (Auditors) So make it valuable for them.

## Segment 5: Conclusions (~5 minutes)

Table-groups will discuss ways to radically simplify and/or automate the documents listed on their canvas. And for each, if the quality expectations described in Segment 4 cannot be achieved, the document will be removed from the canvas and discarded.

### Closing

Table-groups will be invited to list the documents they were able to remove from their canvas. Celebration will ensue!