Planning

Big Up Front Planning – BUFP

* Helps to deliver the **right value** at the **right time**.
* Control the costs
* Manage the risks

Running late on a project using Waterfall style development

* May have to deploy before integration and system tests
* Not good as a lot of things break during integration tests
* May have nothing to deliver as may not be integrated
* Possibility of **most wasted time**

Running late on a project using iterative process

* Better situation than Waterfall-style if have to deploy early because may have some working version of the application. But chance of that is lower than agile because iterations are larger.

Running late on a project using iterative and incremental process

* Best case, as may have the work of completed iterations due to emphasis on **delivering early and often and on delivering real value.**
* May have already deployed some useful part of the code.
* **Least wasted time**

Agile Planning Approach

* Just-in-time/just enough
* Iterative and incremental
* Value-driven.

Software requirement prioritisation

* Prioritising requirements prevents wasted developments.
* Difficult
* Subjective
* Can take a long time
* Solution: assign relative priorities, just need to decide if story x is more important than story y, rather than quantify the priority. Same as prioritising values. Make process quicker.

**Stories** should be **independent** of each other so can choose to work on the **high value stories first.**

Release planning: don’t understand how that works with working on one user story and releasing it.

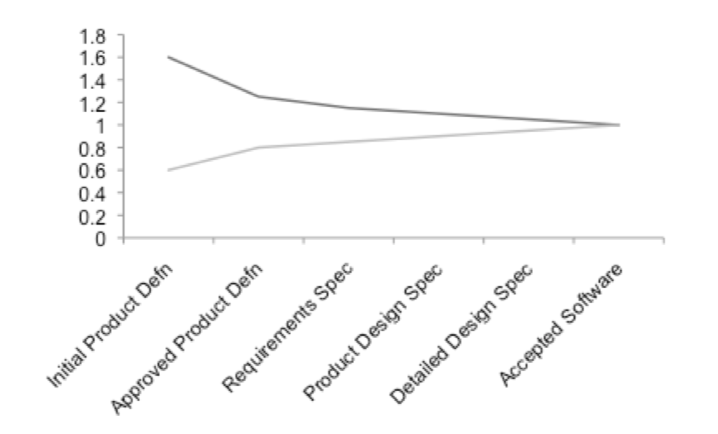
* 2 approaches: **date based and story based.**
* **Date based:** Have deadlines by which we must deliver solutions.
* Need to work out what functionality we can deliver before the deadline.
* Group stories into sets that correspond to a release by some deadline.
* Story based: we have a set of stories that we hope to deliver, and we want to know how long it will take to complete them.

Story Wall

* Visible current status

Estimate the stories

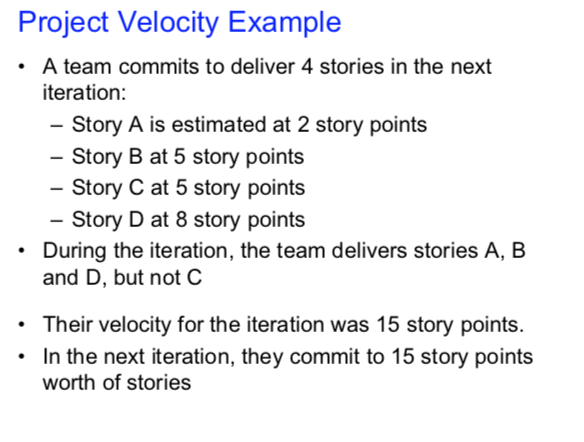
* **Assign the stories to iterations.**
* **Need to know which stories can fit into the iteration size.**
* Software Estimation is challenging
* Possible approach: estimation \* 3



* Shows that estimation of how long the development will take becomes more accurate as time goes on.
* **Traditional approach**: break task into smaller components, estimate these, bring together to estimate large component. This is time consuming and not the most accurate method.
* **Agile Approach**: Duration = Effort / Number of People, estimate effort in terms of story size, estimate relative size, not absolute size. (kinda get it)

Nebulous Units of Time (NUTs)

* Relative allocation of points to stories
* If story A is estimated at 2 story points and story B at 4, that means (only) that B is about twice the size of A
* The number of story points a team delivers in an iteration is called the **project velocity.**
* **Early iterations** are used to learn how many story points they can deliver in one iteration.



* The **predicted velocity** is used for planning.