www.spkaa.com
Ph: 888-310-4540

SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Software Development Using Kanban

Introduction

The SPK development team engages with customers utilizing agile principals. This typically means incremental sprint deliveries of code as part of a scrum model. At a recent engagement, we employed an interesting variation of this called Kanban.

The concept of Kanban has its origins in the grocery store industry with respect to keeping display shelves stocked.

- Typically items are removed from the shelf by a customer.
- At a certain point the grocery clerk notices they're running low on a particular item. The signal is the visible stocking label (or card) which is viewable only when the last item is removed.
- The grocery clerk then must go to the back storeroom to collect more of the same item to restock the shelf.
- However, if the storeroom supply is also low (down to the reorder level), the clerk must then order more of the same item from the manufacturer.

Toyota in the 1940's adopted a similar model to remove the problem of having an excess or insufficient inventory of parts. Their on-demand model made use of 3 bins and note cards:

- One bin for the factory floor
- One bin for the factory store
- One bin for the supplier's store

Each note card contained specifications for one specific part. And the total number of cards created was fixed. i.e. only 100 radiator cap cards for example.

If a customer purchased an item from the supplier, their bin would be reduced by one card for that specific part. Once the supplier bin ran out of cards, their empty bin would go to the factory store for restocking. Once the factory store bin was depleted, they would send their empty bin to the factory floor signaling them it was time to make more of that part. So by keeping an eye on the cards in the bin and knowing that an empty bin was a *signal* to restock or make more parts, a Just-In-Time (JIT) production model was created. The coined label *Kanban* is a combination of the two words *kan* and *ban;* kan meaning visual and ban meaning card.

Most engineers are familiar with Scrum as it relates to software development. So how does kanban apply and what are some of the main differences?

Both scrum and kanban software development models begin with creating a visual representation of all the tasks and the work flow.

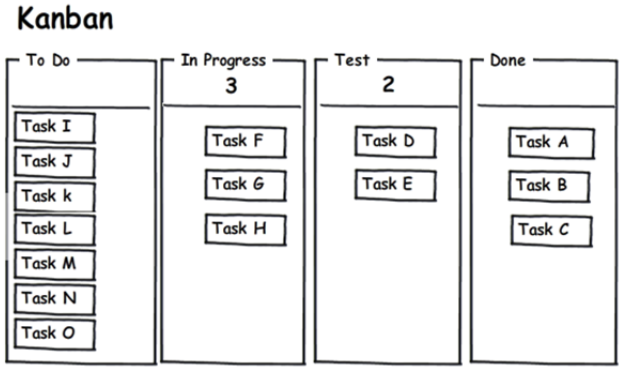
One of the 1st key differences between both models is that kanban limits the amount of work that can happen in any one phase (or column).

www.spkaa.com
Ph: 888-310-4540

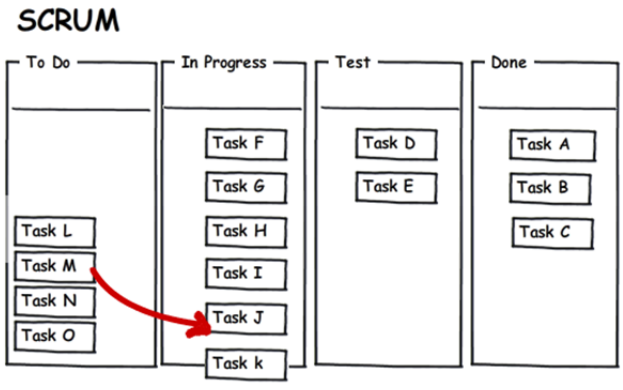
SPK and Associates
900 E Hamilton Ave, Ste.100
Campbell, CA 95008

Just like the car factory worker is not permitted to produce more parts than the total number of cards, no team member may pull in a task if the limit of tasks for that phase has been reached.

In the kanban graphic example below, no more than 3 tasks can be in the "In Progress" phase and no more than 2 tasks can be in the "Test" phase. Measurement of the *cycle time* is defined as the average amount of time it takes one item to go from beginning to end.



In contrast, the Scrum graphic example below shows that team members are permitted to pull in as many tasks as they desire for any phase (as long as it is approved by the Product Owner for the sprint). The team's understanding of how many tasks they can accomplish (velocity) is measured at the sprint level (a time unit – such as 2 weeks).



A second difference between Kanban and Scrum is respect to organization and roles. Scrum is more prescriptive requiring cross functional team membership with roles such as Product Owner and Scrum Master. Kanban does not require this structure and is therefore considered by some easier to introduce into existing company infrastructures.

Kanban has a finer focus on the identification and removal of bottlenecks due to its limitation of tasks per phase. For example, if the *In Progress* items are being prevented from moving to *Test* due to an issue, the flow stops until the issue is resolved.

The concept of cadence is commonly contrasted between Scrum and Kanban. In Scrum the cadence is typically on delivering increments of working software with timeboxes. In Kanban the cadence focuses on the flow of delivering minimal marketable features with no timeboxes.

Ultimately both systems share the premise of the *feedback loop*:

Make a Change
=> Review the Results
=> Learn from it => Make another Change

And both systems strive to make the feedback loop as short as possible one can make corrections quickly.

Carlos Almeida
SPK and Associates
Architect, Software Engineering