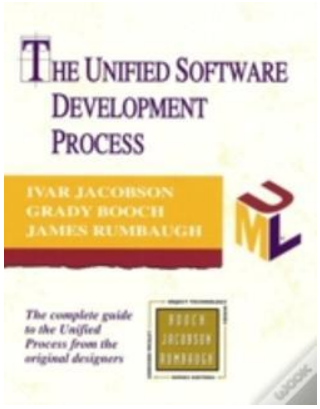
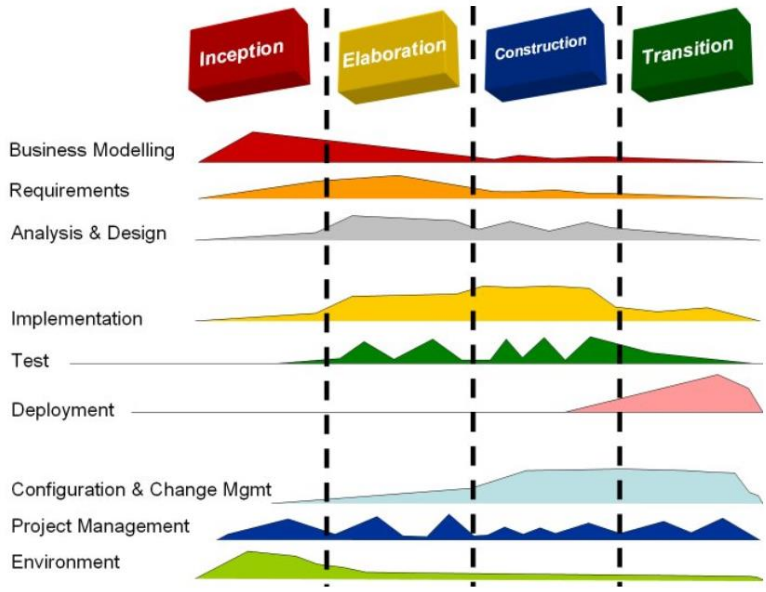
# Unified Process

The first book to describe the unified process was titled **The Unified Software Development Process** and was published in **1999 by Ivar Jacobson, Grady Booch and James Rumbaugh**. There are many variations of the process such as **The Rational Unified Process (RUP),** created by **Rational Software**, a division of IBM, in **2003**.




## Iterative and incremental

The iterative and incremental aspect can be divided into 4 phases.




- **Inception**

The inception phase is the first and should be the least demanding of the 4 phases. Its purpose is to define a scope for the project, to establish objectives and to assess feasibility in function of effort. The team must come up with:

  - A list of requirements, features and main problems.
  - A prototype use-case model.
  - An initial project schedule.
  - A risk assessment.
  - A plan, predicting phases and iterations.
  - A scope.

- **Elaboration**

The elaboration phase is where you develop the groundwork for your construction phase. You assess errors, build the architecture and eliminate the highest risk elements aiming to:

  - Produce a strong, implemented baseline for your system.
  - Create a detailed plan to be used on the construction phase.
  - Ensure that the necessary tools, processes, and standards have been put in place to advance to the next phase.
  - Continue to detect and eliminate the highest priority risks of your project.

By the end of this phase it is expected a functional and executable architecture prototype awaiting to receive the approval from the employers. This is the last step where it is possible to back down with minimal losses.

- **Construction**

The construction phase is the largest of all 4 phases and should consist of building the project using the results from the elaboration phase as a mock-up. Its principles are:

  - It should be flexible as the plan is subject to modification.
  - It should have all the requirements and tools ready to implement the project.
  - It should be assisted with diagrams such as UML to help the team view the whole project.
  - It should take the most time and effort out of all the phases.

- **Transition**

Finally, in the transition phase, the project should be released and the team should hear the feedback from the costumers, making minor changes to the product. Additionally, the team should prepare for the next cycle with tool upgrades and training.

## Architecture-centric

The Unified Process focuses on building the project from an architecture created in the elaboration phase. It's what guides the team during the construction phase, and what allows the project to be so organized and rapidly implemented.

## Risk-conscious

The process gives major importance to eliminating risks early in the project, in the elaboration phase. They should be addressed before implementing the project

## Advantages and disadvantages

In our perspective, this process has the following advantages and disadvantages:

- **Advantages**

    1. **Prevents bad investments -** This methods prevents massive losses by identifying which projects are viable or not in the elaboration phase instead of spending a lot of money only to crash into a roadblock.

    2. **Better execution of the project -** Because of all the preparation when going into the actual construction phase, the team of developers is much more aware of the steps it has to take and which will be the main hindrances to the project.

    3. **Feedback maters** - Following the Unified Process methodology the consumers opinion is contemplated in the final phase. The team should hear the costumers opinion and patch up what is necessary to make the product user friendly and correct any user related bugs.

- **Disadvantages**

    1. **Saves money but costs time -** While it can prevent great losses it also increases the base project cost by adding a lot of hours dedicated just to preparation and risk assessment instead of starting the project right away. Companies take this time into consideration and sometimes prefer to make a gamble.

    2. **The plan never survives battle** - All the thorough preparation can lead into a false sense of security. Some bugs can not be predicted and some of those can not be solved on the go possibly stalling the project or even cancelling it.

## Sources:


- Scott W. Ambler and Larry L. Constantine. 2016. [The Unified Process Inception Phase](http://www.ambysoft.com/books/inceptionPhase.html).
- Sinan Si Alhir. 2002. [Understanding the Unified Process (UP)](http://www.methodsandtools.com/archive/archive.php?id=32).
- RationalSoftware. 1998. [Rational Unified Process](https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf).
- http://www.informit.com/articles/article.aspx?p=24671&seqNum=5

# KANBAN

## History of Kanban

The first Kanban system was developed in the 1940's by a Japanese industrial engineer and businessman named **Taiichi Ohno** who worked for Toyota Motor Corporation.

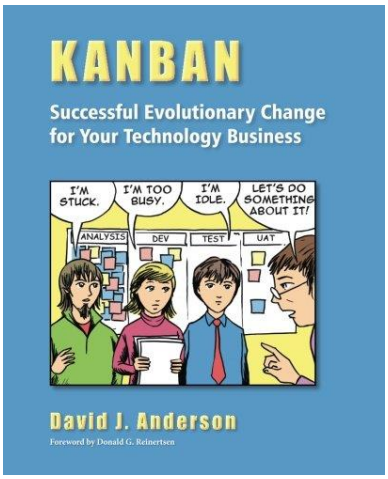

Taiichi Ohno

The Kanban System developed by these Japanese engineer was invented as a simple planning system which aim was to supervise work and inventory at every stage of production. It had the purpose to fight the inadequate productivity and efficiency in industrys and wanted to achieve higher throughput with lower delivery lead times.

The result that Ohno wanted was to control the entire value chain, starting in the supplier and ending on the costumer avoiding in this way supply disruption and overstocking of goods at various stages of the manufacturing process.

The first person to apply this concept to software development and knowledge work in general was **David J. Anderson** in **2004**, while developing an project at Microsoft. By studying and perfecting the works of Edward Demmings, Eli Goldratt, Peter Drucker and others, Anderson had as result the Kanban Method, directionally related to pull systems, queuing theory and flow.

"The two pillars of the Toyota production system are just-in-time and automation with a human touch, or autonomation." – **David J. Anderson**



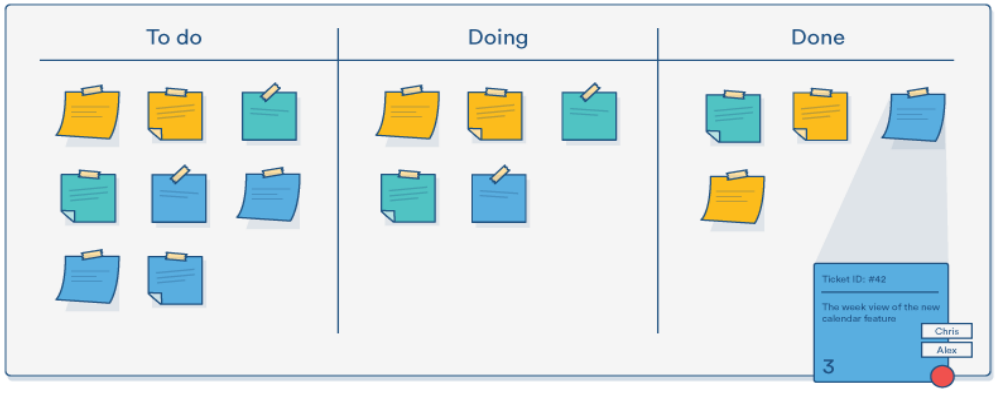
## The Kanban principles

The Kanban Method in building software consists in gradually improving the development.

"Kanban is not a software development lifecycle methodology or an approach to project management. It requires that some process is already in place so that Kanban can be applied to incrementally change the underlying process." – **David J. Anderson**

## The Kanban Board

This method is based in a visualization, using as a tool a Kanban board.




The observation of this board allows a better work and workflow management, providing advises limiting work in progress, permiting the reducion of the waste that results from multitasking and context switching and exposing the operational problems, stimulating the collaboration to improve the system.

As we can see in the board, it is divided in three categories:

- **To Do**: Tasks to work on
- **In Progress**: Tasks currently active
- **Complete**: Completed tasks

The path to execute these categories is starting in the **To Do** category, proceding to **In Progress** and ending at **Complete**. This is the basic path, because a more complex project would obviously require a more complex path with more categories, for example "**Planned**" or "**In Testing**".

The Kanban team that is working on the project meets the needs of the costumer and works on them. Usually this team is not large and has no strict roles, because these roles are defined by the needs in the project, but we can affirm that a typical agile team includes analysts, project managers and testers. Two important roles we can highlight are the Service Delivery Manager and the Service Request Manager.

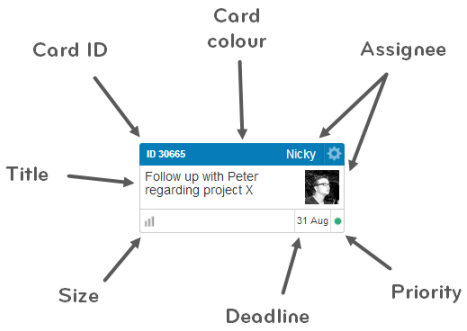
## The Kanban Cards

As we can see in the image of the board, each column has Kanban cards in it. They are simply written messages that indicate the need to replenish a specific component, representing a task.

Some fields of this cards at the simplest level are for example:

- ID: The unique identifier
- Assignees: Persons working on the task
- Description: What the task is.
- Type: The type of the task.
- Timeframe: Time expected that the task will take
- Blocked Indicator: Whether is or not a blocked card

This cards are a fundamental part of the agile Kanban process.




## Blocked Cards

These type of cards are usually indicated with a red marker at the corner of the corner, meaning that the progress on the card is suspended until another action is taken.

## Practices

The Kanban principles are fulfilled by six general practices:

1. **Visualization of the work flow** - Creation of the board according to the complexity of the work

2. **Limiting Work on Progress** - Encourages the persons who are working to complete the task at hand before takin up another more recent work, creating in this way more capacity in the system, so new work can be pulled in by the team - 'Pull-system'

3. **Flow Management** - Control of the workflow and the work status at every stage

4. **Making Process Policies Explicit** - Process rules that explain what to do at each step

5. **Using feedback loops** - The Kanban method encourages and helps the implementation feedback loops or the lack of it

6. **Collaborative or experimental evolution** - By using this agile method, we are submitted to an evolutionary improvement process which helps to adopt small changes and gradually improve at a pace and size easy to handle

## Advantages and Disadvantages of the uses of Kanban Method

As every software process, this method as Pros and Cons:

- **Advantages**

### 1. Allows for Flexibility

Despite the use of basic concepts is very flexible to implement the Kanban methodology by allowing the modification of the board, cards and respective content.

### 2. Forces Event-Driven Workflow

Everything is based on cards that represent tasks and their progress across the development stages on the board, turning this method entirely event-driven and allowing the team to constantly adapt as the cards turn prominent while other can be ignored

### 3. Reduction of the waste

Kanban encourages not only transparency but also an agile workflow permitting that the team stay abreast of the components progress in the project.

- **Disadvantages**

### 1. Complexity Potential

The possibility of the creation of a board over-engineered and complicated is increased according to the complexity of the project with the risk that the created system is too confusing to parse and use.

### 2. Possible Bottlenecks

Not planning for and dealing with blocked cards can cause hindrances waiting on one particular card or component to be done. Kanban reveals a particular danger because its focus on cards makes scheduling and milestoning more difficult.

### 3. Required Constant Board Monitoring

The Kanban board must be constantly monitorized to assure that the cards will not expire.

## Attachment

To help to visualize this process, we recommend watching this video:

- https://www.youtube.com/watch?v=R8dYLbJiTUE

## Sources


- https://www.digite.com/kanban/what-is-kanban/
- https://airbrake.io/blog/sdlc/kanban
- https://en.wikipedia.org/wiki/Kanban_(development)
- https://www.toyota-global.com/company/vision_philosophy/toyota_production_system/just-in-time.html
- https://leankanban.com/emerging-roles-in-kanban/