1. **Discuss the prototyping model. What is the effect of designing a prototype on the overall cost of the project?**

Ans.

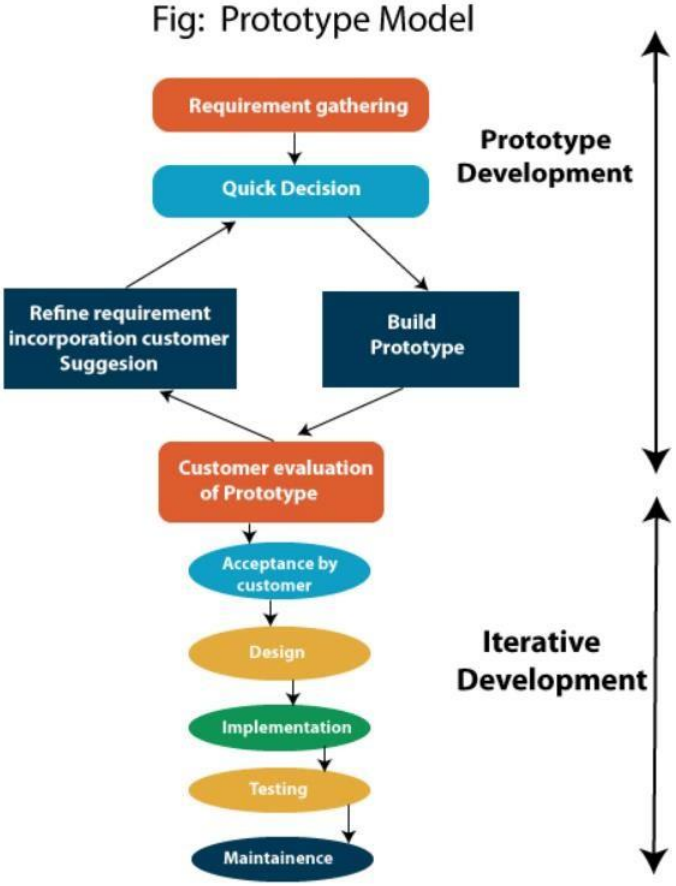
A. Prototyping Model:

The prototype model requires that before carrying out the development of actual software, a working prototype of the system should be built. A prototype is a toy implementation of the system. A prototype usually turns out to be a very crude version of the actual system, possibly exhibiting limited functional capabilities, low reliability, and inefficient performance as compared to the actual software. In many instances, the client only has a general view of what is expected from the software product. In such a scenario where there is an absence of detailed information regarding the input to the system, the processing needs, and the output requirement, the prototyping model may be employed.

B. Steps of prototyping model:

i. Requirement Gathering and Analyst
ii. Quick Decision
iii. Build a Prototype
iv. Assessment or User Evaluation
v. Prototype Refinement
vi. Engineer Product

Fig: Prototype Model



C. The effect of designing a prototype on the overall cost of the project
Prototyping may have some initial costs of development, but it reduces the overall budget by helping your product to be free of the errors or glitches that could have occurred if the idea was made from scratch without any prior user testing. Furthermore, prototyping also helps to understand the intrinsic flaws, shortcomings, and drawbacks that can be improved during the product development process. If the prototyping process is ignored completely, it might result in the restructuring and redesigning of the entire product after spending all your resources on its development. So, the effect of designing a prototype on the overall cost of a software project is to reduce the additional costs of restructuring and reframing it after its full-fledged development- which might cost a fortune.

2. Compare iterative enhancement model and evolutionary process model.

Iterative Enhancement Model	Evolutionary process Model
1. Based on an iterative approach.	1. Based on an incremental and iterative approach.
2. Release product after each cycle.	2. Does not release product after each cycle.
3. Less flexible for work	3. More flexible for work.
4. High expertise is required.	4. Medium expertise is required.
5. Resource control can be done	5. Resource control cannot be done.

3. As we move outward along with process flow path of the spiral model, what can we say about software that is being developed or maintained.

As work moves outward on the spiral, the product moves toward a more complete state and the level of abstraction at which work is performed is reduced (i.e., implementation specific work accelerates as we move further from the origin).

Spiral Model:
Spiral model is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.
The risk-driven feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

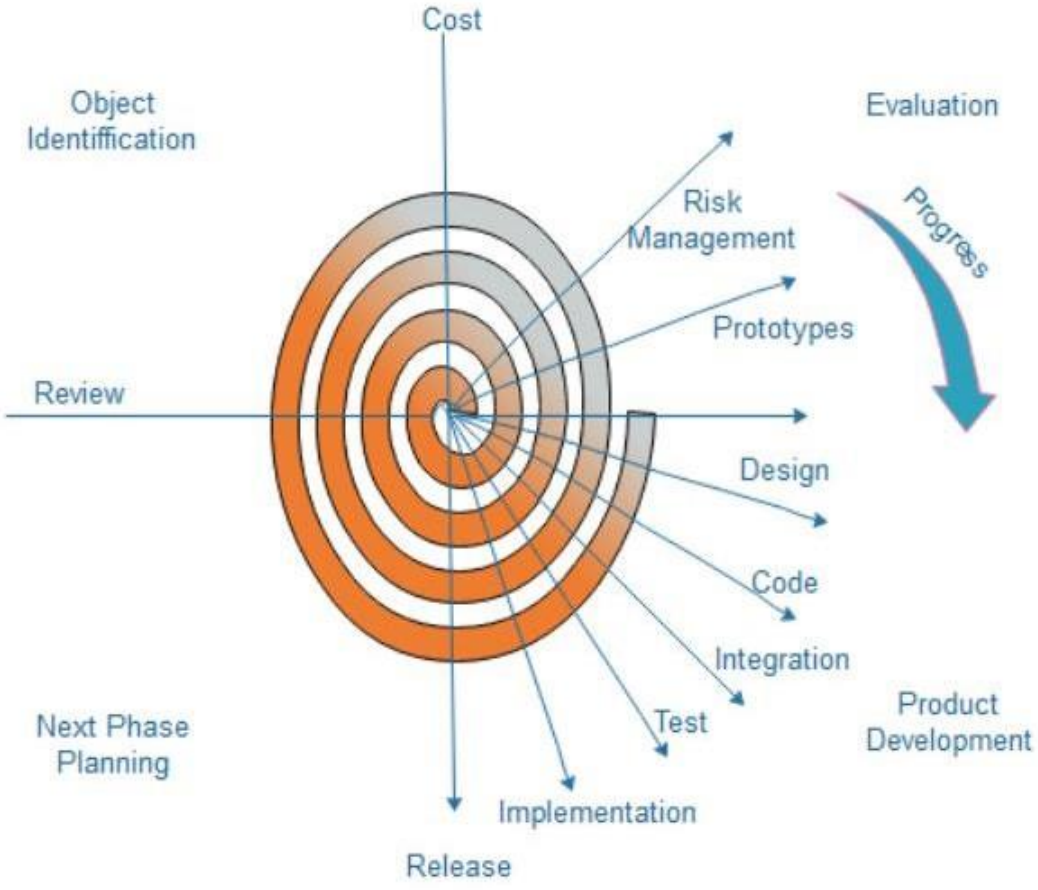
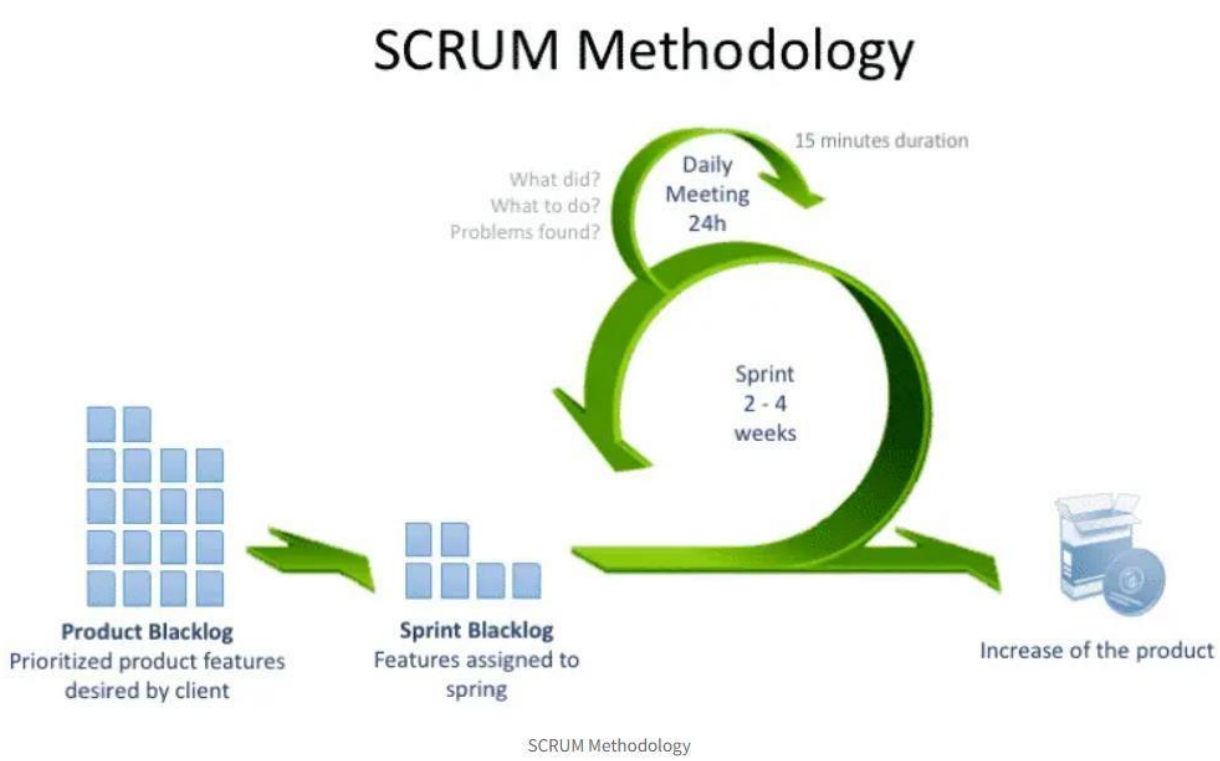


Fig. Spiral Model

4. Explain the Scrum Agile methodology.

One of the most popular agile methodologies in use today. Scrum is a lightweight software development methodology that focuses on having small time-boxed sprints of new functionality that are incorporated into an integrated product baseline. Scrum places an emphasis on transparent customer interaction, feedback and adjustments rather than documentation and prediction.
Instead of phases, Scrum projects are broken down into releases and sprints. At the end of each sprint, you have a fully functioning system that could be released.

With scrum projects, the requirements for the project do not have to be codified up-front, instead they are prioritized and scheduled for each sprint. The requirements are composed of ‘user stories’ that can be scheduled into a particular release and sprint.



SCRUM Methodology

1. Flexibility and adaptability – since the requirements (consisting of features, user stories and other backlog items) are deliberately kept high-level at the start, with the ability to redefine and replan after each release and sprint, you have the flexibility to adapt the requirements from feedback from the customer and end users 'on the fly'.
3. Creativity and innovation – Scrum lets integrated teams of developers, designers, testers, and business/functional experts all work together rather than working in individual 'silos'. This cross-discipline collaboration fosters creativity and innovation.
4. Lower risks and costs – a study found that the majority of features delivered in traditional waterfall projects were never used or needed by customers, due to scope creep and feature bloat. With Scrum, you are ruthlessly pruning the release backlog after each sprint so such features (and their associated costs) are quickly weeded out. The short-time boxes reduce the likelihood of risks materializing during the sprint and thus addressing increment delivery.
5. Quality improvement – With engineering approaches such as continuous exploration, continuous integration, continuous delivery, test-driven-development, DevOps and relentless test automation, Scrum yields much higher-quality code, products and process.
6. Organizational synergy – Scrum lets team members from different disciplines harness their different skills in a cross-cutting product development ethos, so that the finished product integrates the best skills in the organization not just what's written on their job title.
7. Employee satisfaction – With the flexibility to adjust scope and still deliver results, the problem of delivering a fixed scope, fixed-schedule project with a team goes away. Instead of long nights and frantic replanning and retesting of functionality, the team has a sustainable operating pace that makes employees proud of their work and not fighting to deliver against impossible deadlines.

5. Customer satisfaction – Once of the challenges of software development for customers is that they cannot often appreciate the functionality of a system until they see it. By delivering working functionality in small increments, customers can see what they are getting incrementally and be confident that it is what they will need.

5. Explain the utility of Kanban CFD reports.

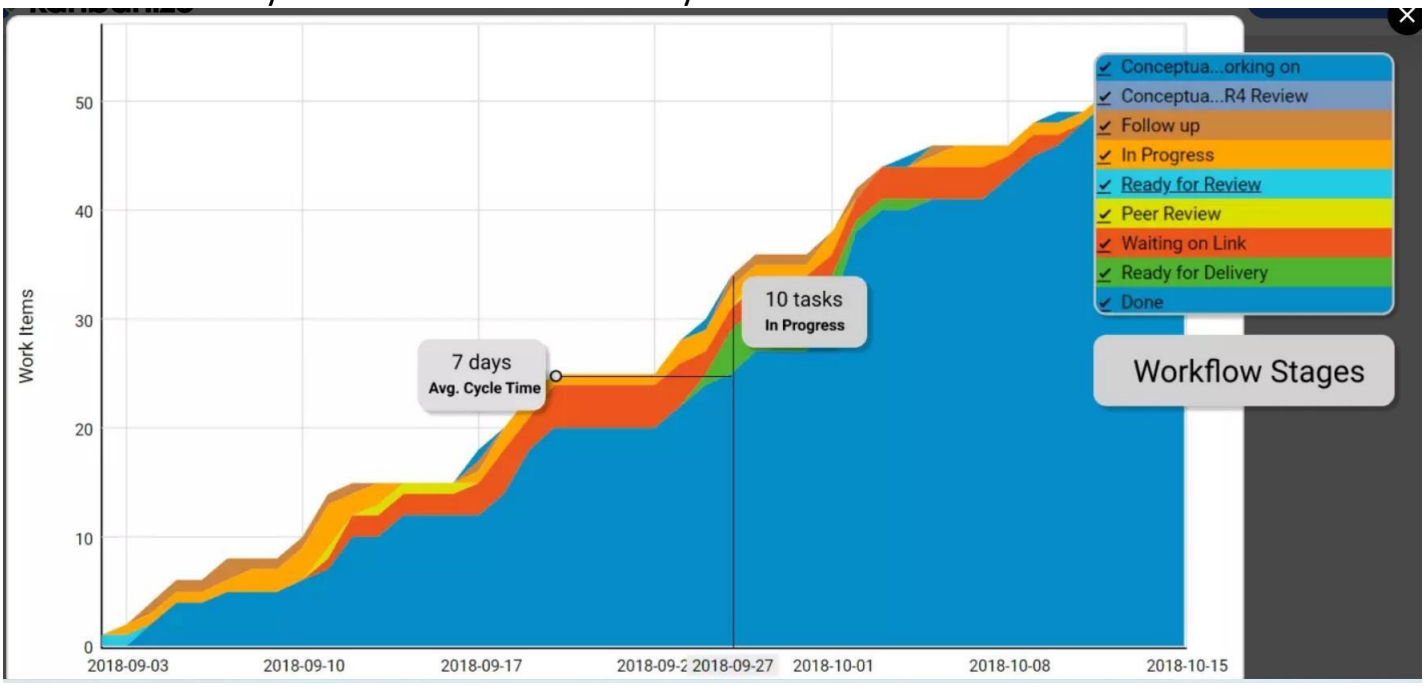
The cumulative flow diagram (also known as CFD) is one of the most advanced Kanban and Agile analytics charts. It provides a concise visualization of the three most important metrics of your flow:

- Cycle time
- Throughput
- Work in progress

Its main purpose is to show you how stable your flow is and help you understand where you need to focus on making your process more predictable. It gives you quantitative and qualitative insight into past and existing problems and can visualize massive amounts of data.

A. How To Read a Cumulative Flow Diagram

The chart tracks the total number of work items in the columns of the "In Progress" section on your Kanban board each day.



The horizontal axis of the CFD represents the time frame for which the chart is visualizing data. The vertical axis shows the cumulative number of cards in the workflow at various points in time.

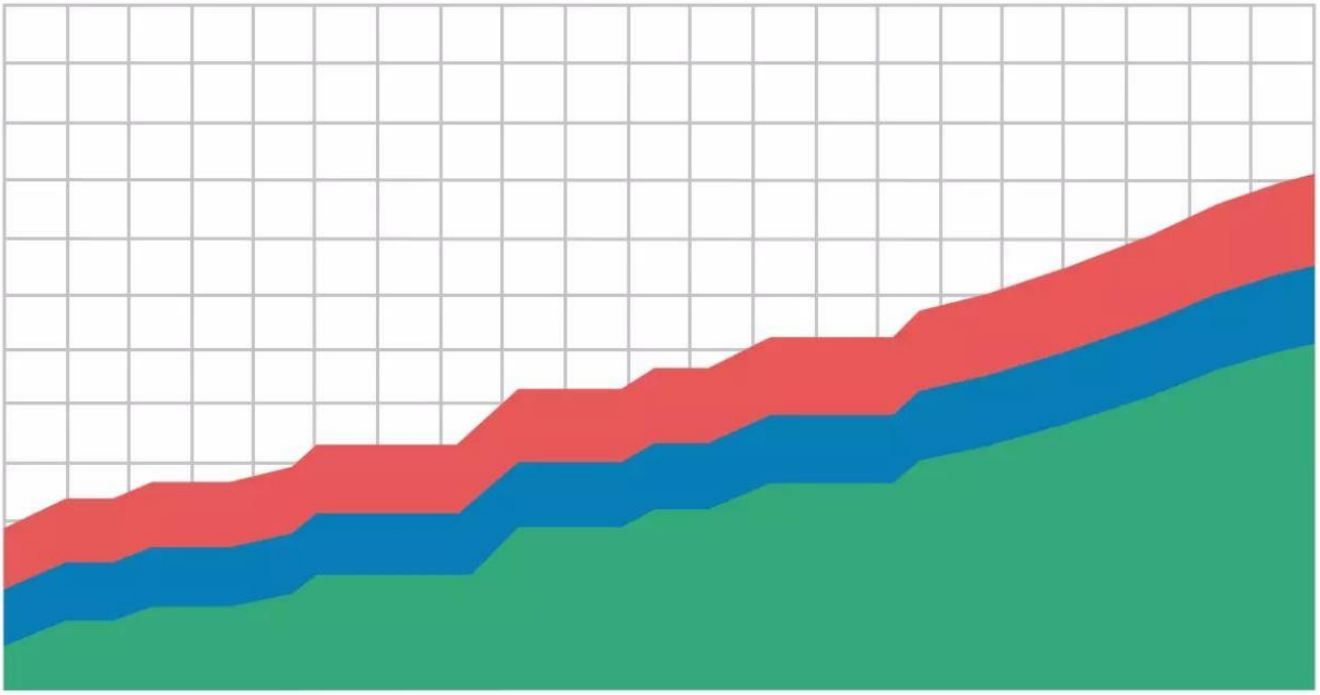
The differently colored bands that divide sections of the upward flow are the different stages of your workflow as they appear on the Kanban board itself. The bands always go up or sideways in accordance with the number of assignments that go through your process.

The top line of each band on the cumulative flow chart represents the entry point of tasks in the respective stage of your Kanban board, while the bottom one shows when it leaves it. If a line becomes flat, nothing arrives in the corresponding stage, or nothing is leaving it.
Using a CFD, you can get an idea of how long your tasks' approximate cycle time is. This is possible by measuring the horizontal distance between the top line of the first stage on the cumulative flow diagram and the bottom line of the last "in progress" stage.
The number of days/weeks/months that have passed is the approximate average cycle time of your team's assignments for the time frame.
The distance between the lines of a CFD will show you the problems of your workflow.

B. Understanding the data on a CFD Chart
You can spot whether your process is stable in just a single glance by looking at how the top and the bottom line of each band in your cumulative flow diagram are progressing.

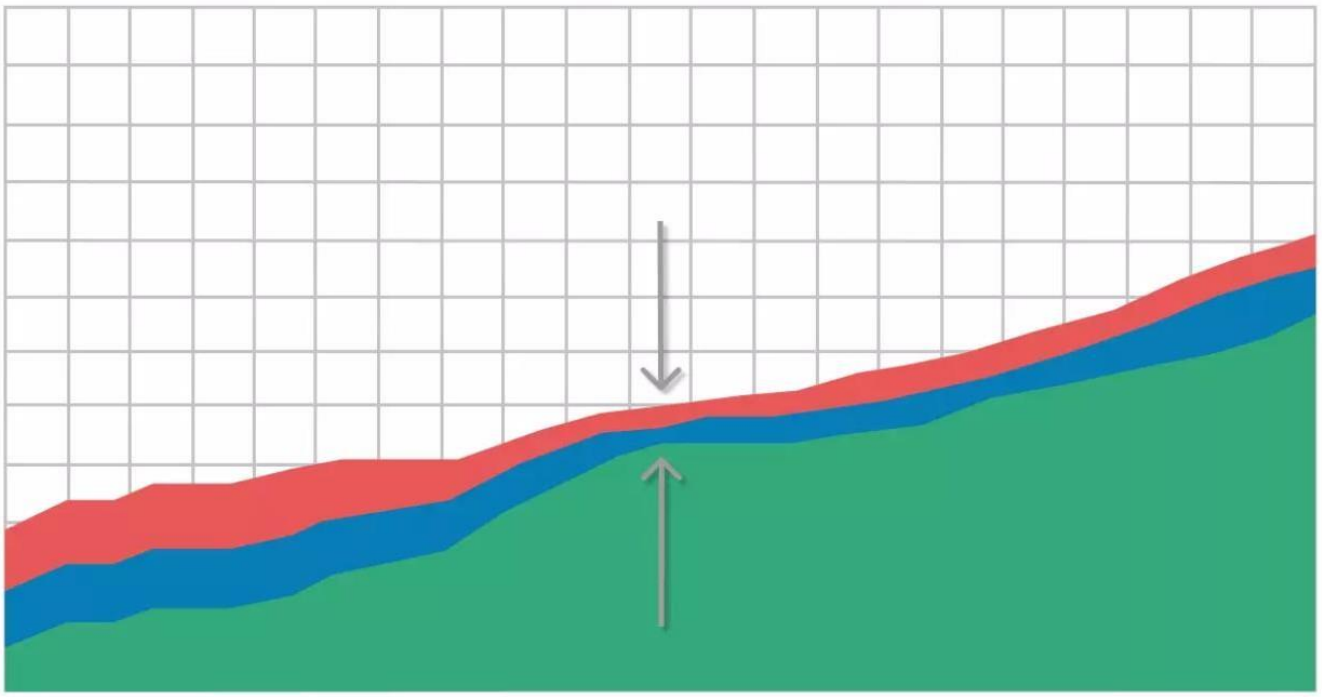
There are three common scenarios:

i. The Bands are Progressing in Parallel



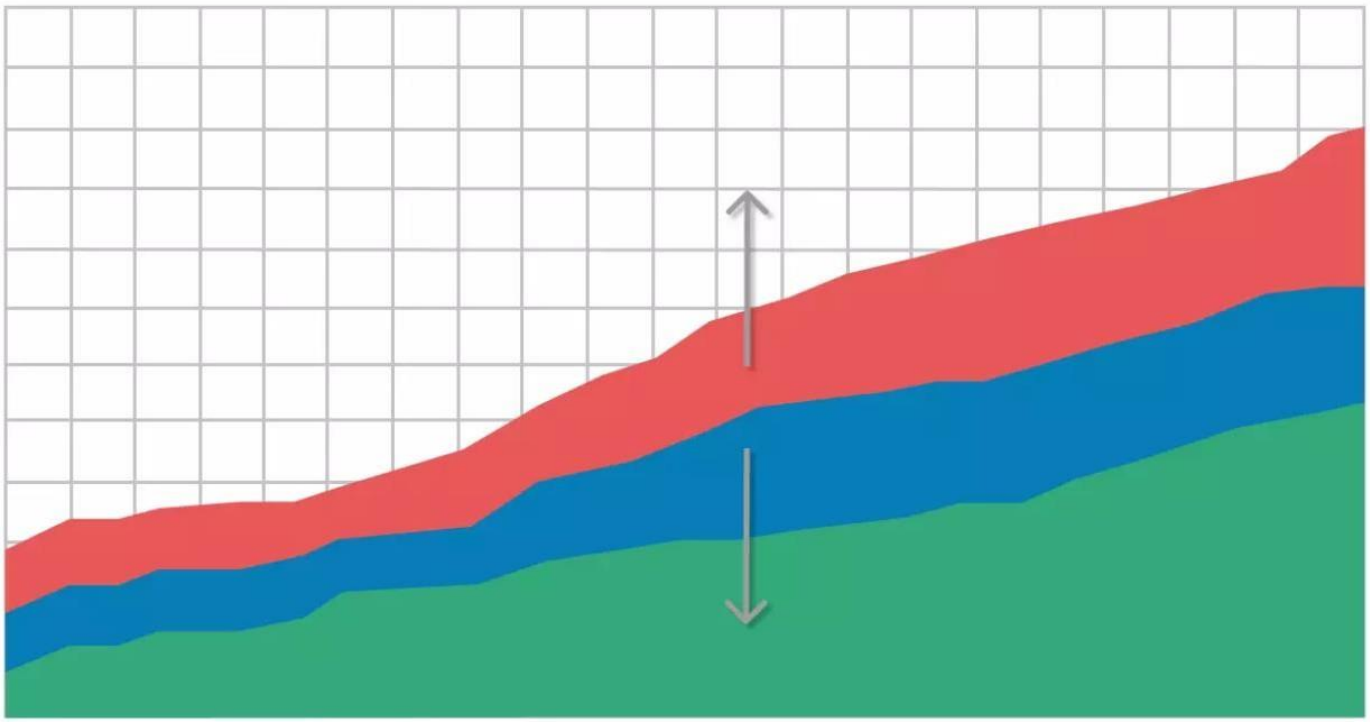
This means that your throughput is stable, and new tasks are entering your workflow in parallel to those that are leaving it. This is the ideal outcome and shows that you can focus your efforts on shortening your assignments' cycle times.

ii. A Band is Rapidly Narrowing



If a band on your CFD is continuously narrowing, that means that the throughput of the stage it represents is higher than the entry rate. This is a sign that you've got more capacity than you really need at this stage, and you should relocate it to optimize the flow.

iii. A Band is Rapidly Widening



Whenever this happens on a cumulative flow diagram, the number of cards that enter the corresponding stage on the Kanban board is higher than the number of assignments leaving it. It is a common problem caused by multitasking and other waste activities that don't generate value.

There are many possible actions to resolve this issue. However, if this is not generated by a dependency on external stakeholders, you should reconsider the existing WIP limits on your Kanban board and focus on finishing tasks that are in progress before starting new ones.