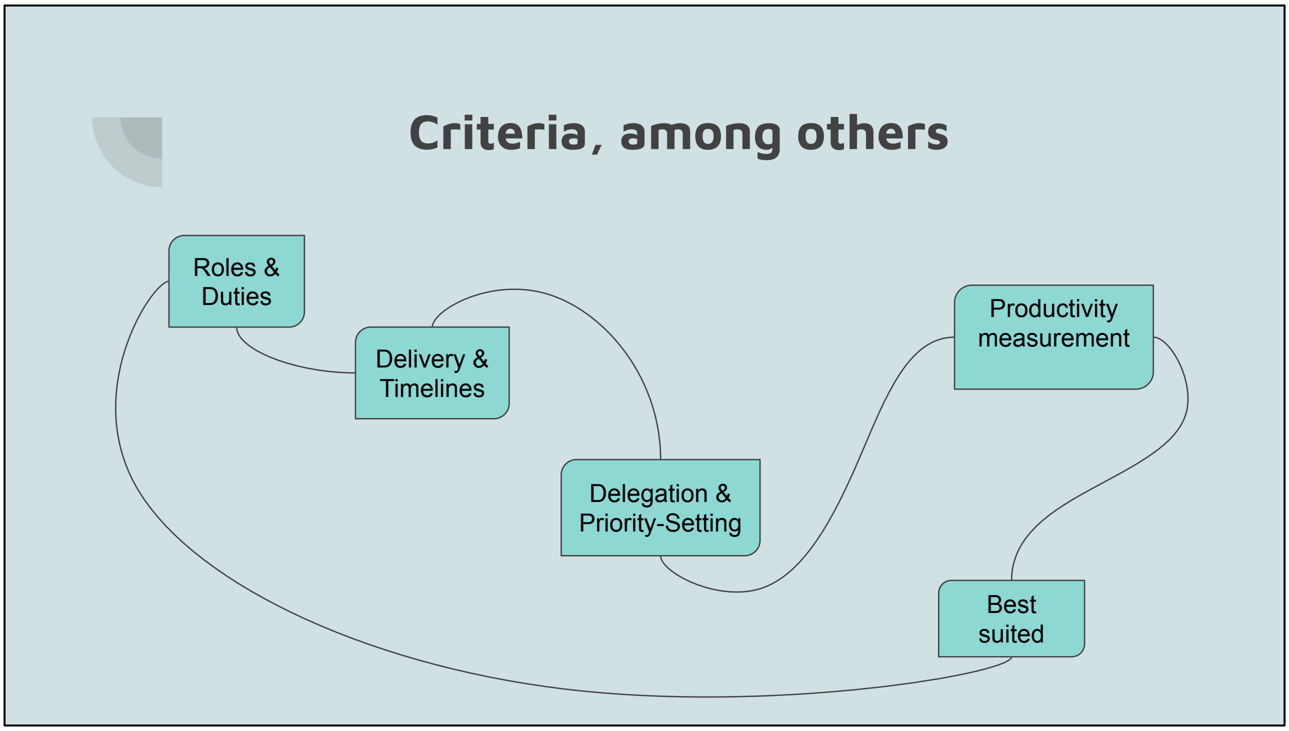
Kanban

Scrum & Waterfall

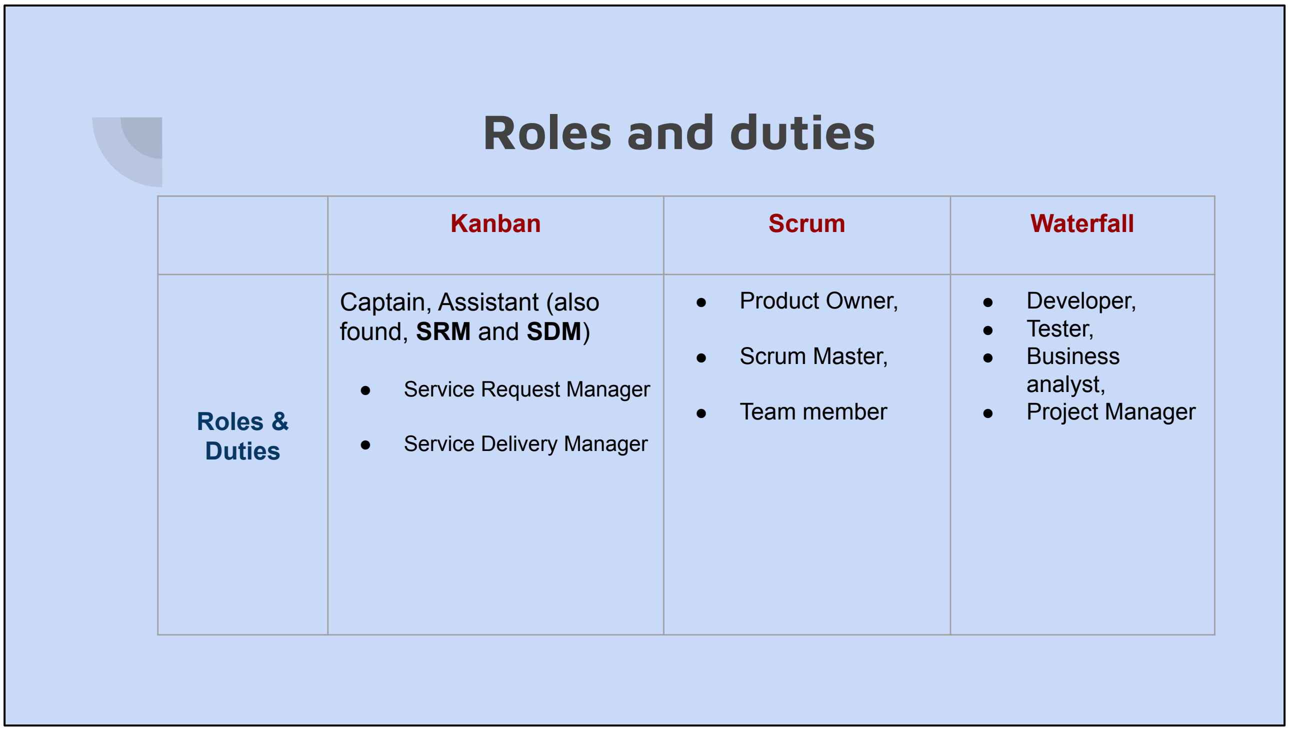
Similarities and Differences

In order to gain clarity let's start by pointing up how differently each system deals with the same kind of issues, and in order to do that let's set some criteria



Criteria, among others

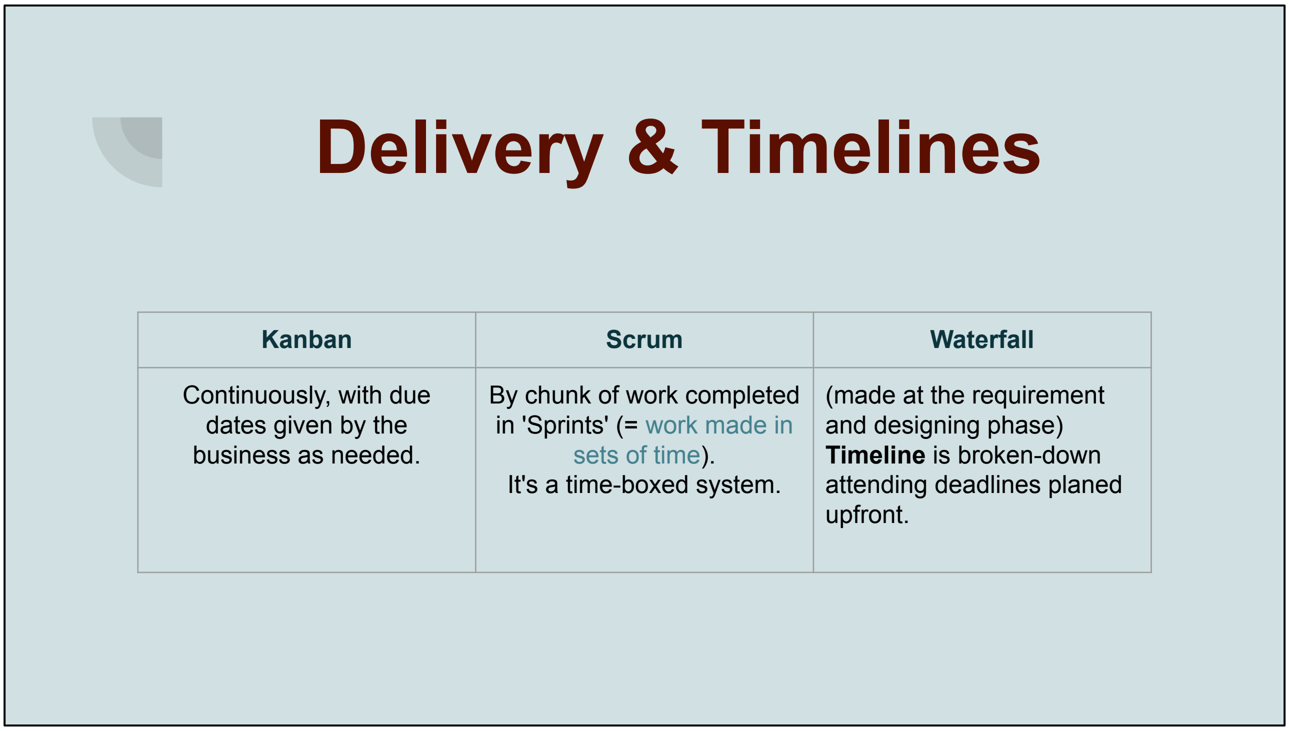
It seems better to think of Agile as a movement where e.g. Scrum and Kanban are applications and Waterfall, a kind of method by default.
Now let’s see how each one deals with roles, duties, delivery, timelines, work delegation, priority-setting, and productivity measurement to have an idea about the work context each one could suit better with.



Roles and duties

	Kanban	Scrum	Waterfall
Roles & Duties	Captain, Assistant (also found, **SRM** and **SDM**) • Service Request Manager • Service Delivery Manager	• Product Owner, • Scrum Master, • Team member	• Developer, • Tester, • Business analyst, • Project Manager

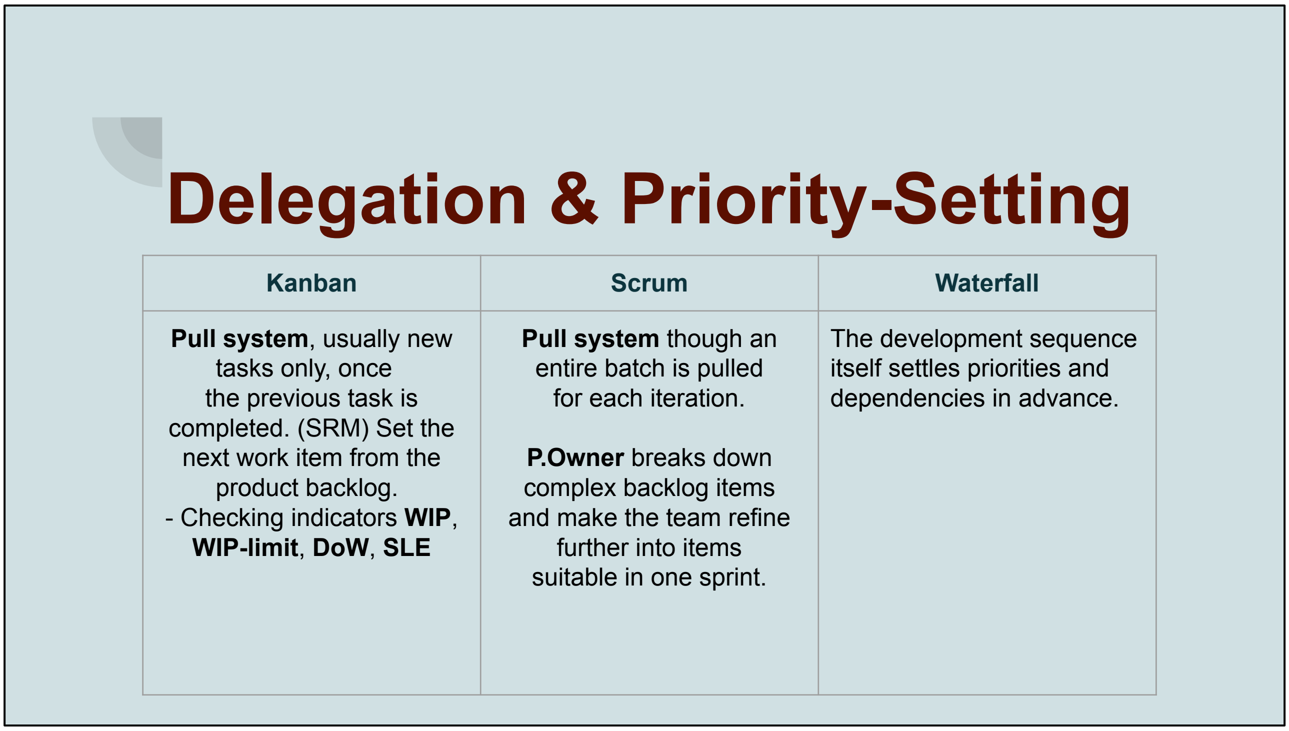
The first thing someone is likely to wonder when working in a team environment is who will do what and how they would be qualified to do it. here is a slide on how each method solves this issue. Although these two roles are accepted in Kanban, they are not mandatory, but a kind of transition, as I have found, to help when migrating from a more rigid environment to a kanban approach (based on self-organisation). And self-organisation against structures more settled.



Delivery & Timelines

Kanban	Scrum	Waterfall
Continuously, with due dates given by the business as needed.	By chunk of work completed in 'Sprints' (= work made in sets of time). It's a time-boxed system.	(made at the requirement and designing phase) **Timeline** is broken-down attending deadlines planed upfront.

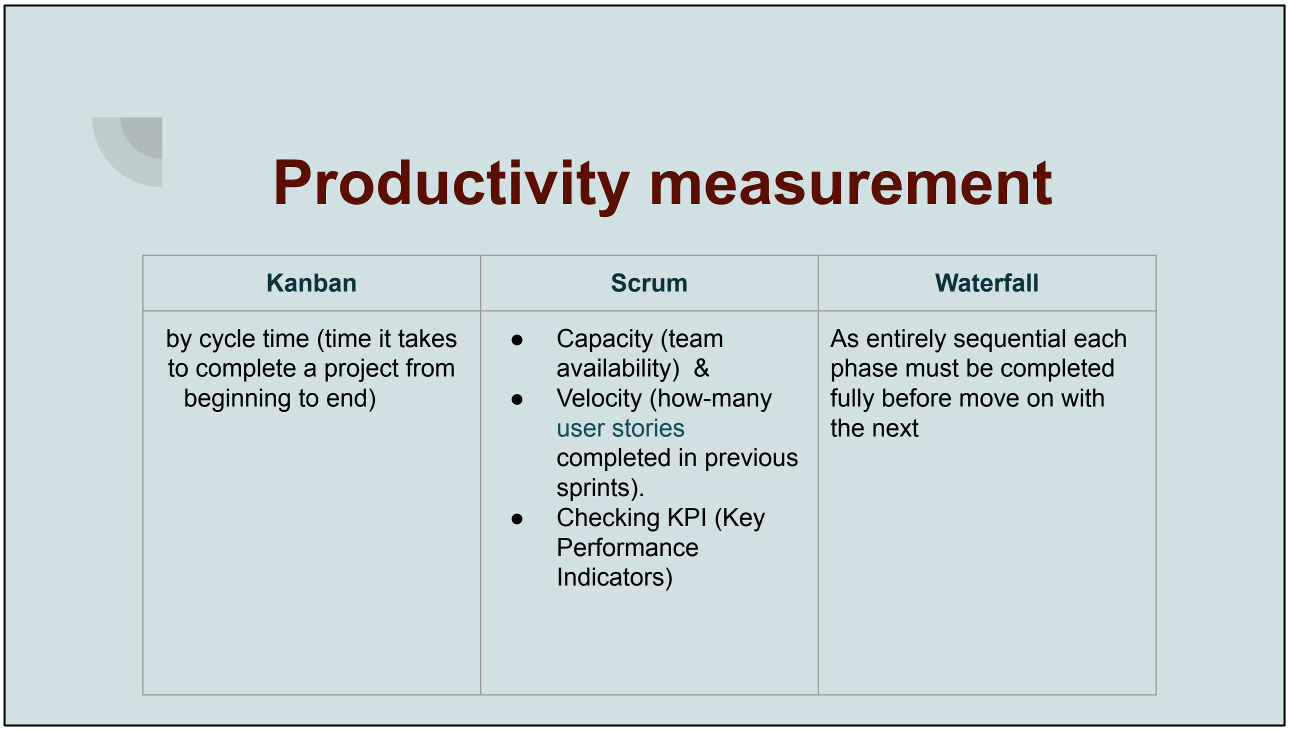
As in kanban, the first focus is the contact with the customer, taking up work (from requirements) and working on it, are the only 2 steps that rule. Each business has different needs, and probably not all businesses will fit in entirely. When the project requires namely more complex testing, dividing it and testing whether it qualifies to merge (a time setting like Scrum, is probably more convenient). In Waterfall, each step depends on the previous stage and the stage's deadlines are defined upfront during the designing phase.



Delegation & Priority-Setting

Kanban	Scrum	Waterfall
Pull system, usually new tasks only, once the previous task is completed. (SRM) Set the next work item from the product backlog. - Checking indicators **WIP**, **WIP-limit**, **DoW**, **SLE**	**Pull system** though an entire batch is pulled for each iteration. **P.Owner** breaks down complex backlog items and make the team refine further into items suitable in one sprint.	The development sequence itself settles priorities and dependencies in advance.

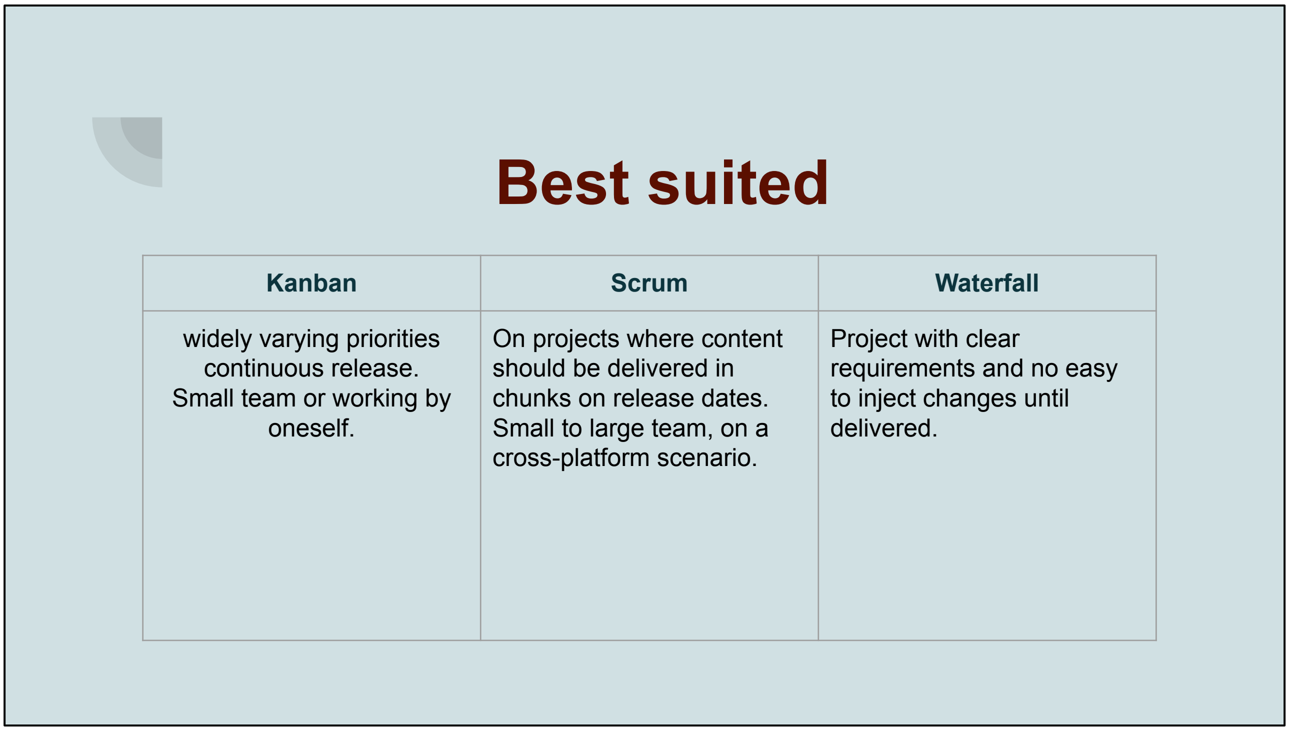
DoW=Definition of Work in kanban is a keystone, WIP= Work in Progress, Work-limit, the amount of workload the team is able to deal with, all these are tools that help to accomplish and evaluate tasks. Both Kanban and Scrum
make use of a pull system to take chunks of work but with some differences, new task only in kanban and accumulative at each interaction in Scrum. Waterfall keeps a kind of mounting software-parts



Productivity measurement

Kanban	Scrum	Waterfall
by cycle time (time it takes to complete a project from beginning to end)	● Capacity (team availability) & ● Velocity (how-many user stories completed in previous sprints). ● Checking KPI (Key Performance Indicators)	As entirely sequential each phase must be completed fully before move on with the next

Focusing on a little bit at a time will make a difference to the project as a whole. In kanban, this is evaluated on a project-by-project basis and would give the team info about its own evidence-based productivity. In Scrum, this feedback is taken by sprint so that performance and productivity insights stay alive, and are part of the now of the project. As keeping the timeline is key in Waterfall, a way of having a measure of the team productivity is to check the ratio between work planned and work delivered.



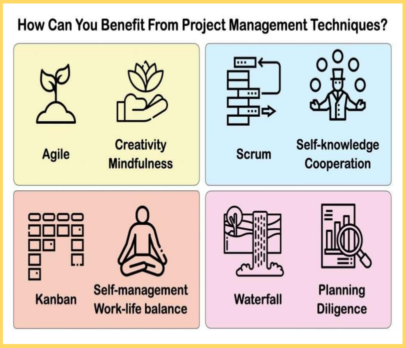
Best suited

Kanban	Scrum	Waterfall
widely varying priorities continuous release. Small team or working by oneself.	On projects where content should be delivered in chunks on release dates. Small to large team, on a cross-platform scenario.	Project with clear requirements and no easy to inject changes until delivered.

Once iteration with customer and changes are defined it is easier to decide working under a waterfall model or agile.
However things are some more complicated when is about deciding in between using Scrum or Kanban.

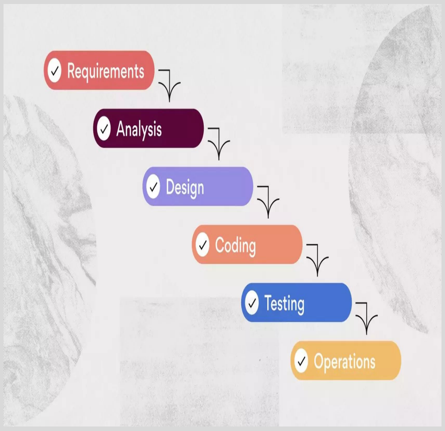
Similarities: kanban, Scrum, Waterfall

Are they really different after all?


How Can You Benefit From Project Management Techniques?
Agile
Creativity Mindfulness
Scrum
Self-knowledge Cooperation
Kanban
Self-management Work-life balance
Waterfall
Planning Diligence

This a curios chapter on the subject as we are going to summarise next. The first question arises easily, are they different methods, after all? There are as I have found out in doing this report several approaches to this question but I have got a special interest in the following chronicle.

"... but the implementation described [] is risky and invites failure."



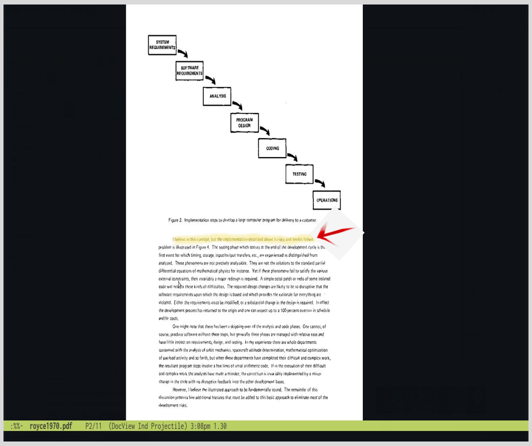


Figure 2. Implementation steps to develop a large computer program for delivery to a customer.

I believe in this concept, but the implementation described above is risky and invites failure. The problem is illustrated in Figure 4. The testing phase which occurs at the end of the development cycle is the first event for which timing, storage, input/output transfers, etc., are experienced as distinguished from analyzed. These phenomena are not precisely analyzable. They are not the solutions to the standard partial differential equations of mathematical physics for instance. Yet if these phenomena fail to satisfy the various external constraints, then invariably a major redesign is required. A simple octal patch or redo of some isolated code will not fix these kinds of difficulties. The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and one can expect up to a 100-percent overrun in schedule and/or costs.

One might note that there has been a skipping-over of the analysis and code phases. One cannot, of course, produce software without these steps, but generally these phases are managed with relative ease and have little impact on requirements, design, and testing. In my experience there are whole departments consumed with the analysis of orbit mechanics, spacecraft attitude determination, mathematical optimization of payload activity and so forth, but when these departments have completed their difficult and complex work, the resultant program steps involve a few lines of serial arithmetic code. If in the execution of their difficult and complex work the analysts have made a mistake, the correction is invariably implemented by a minor change in the code with no disruptive feedback into the other development bases.

However, I believe the illustrated approach to be fundamentally sound. The remainder of this discussion presents five additional features that must be added to this basic approach to eliminate most of the development risks.

```
:%%-  royce1970.pdf    P2/11  (DocView Ind Projectile) 3:08pm 1.30
```

It happens that the waterfall concept first discussed in a paper dated 1970 by Winston W. Royce for an event (known as an IEEE WESCON) was really a caveat immediately after a proposal on improvement on development methodologies. Actually, neither the term 'waterfall' was used in the article, nor he did advocate for it exactly, he rather said figuring out a process like the one known nowadays as 'Waterfall' "is risky and invites failure". In the slide a page where you can find the mentioned quote.

So according to this, Agile (e.g. Scrum, Kanban) share similar approach, really

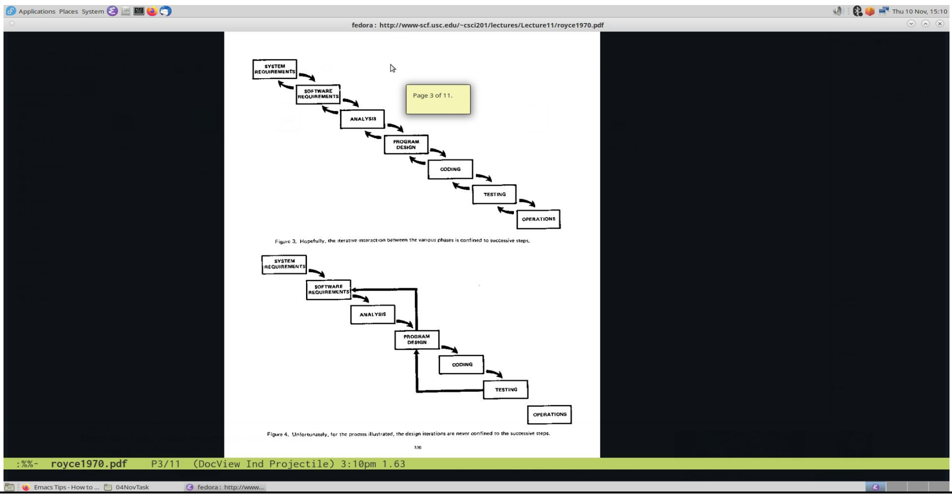


Figure 3. Hopefully, the iterative interaction between the various phases is confined to successive steps.

Figure 4. Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.

330

```
:%%-  royce1970.pdf    P3/11   (DocView Ind Projectile) 3:10pm 1.63
```

Unless, indeed unless certain corrections are introduced. And what corrections are they? Here is the point I wanted to come to, corrections that have to do with avoiding confining stages to their next step in the sequence and or keeping iterations incrementally between each step with the previous. Exactly the reasoning behind the innovations that the Agile methodology has come up to. **>>**Quod erat demonstrandum**<<**