Iterative versus high ceremony software development

How to solve the paradox of a high iterative process with high ceremony level

Renaud FLORQUIN
FloConsult SPRL

Isabelle LECLERCQ
FloConsult SPRL

Developing using iterative or incremental approach offers a lot of benefits but the development sometimes requires a high level of ceremony (i.e. documents, formal reviews, …). This paper shows the influence of ceremony on the iterative development and how we can produce the required level of ceremony keeping the benefits of an high iterative process.

Abstract

In this article, we will first focus on one of the major key practices of agile development: the iterative development.

Afterwards, we will analyze a controversial aspect of the process: the ceremony level and we will discuss why it can be important or sometimes required.

Based on those two variables (the iteration level and the ceremony level), we will classify several well-known processes (i.e. process map).

Then, using this process map, we will discuss about the independence of those axes: the iteration level and the ceremony level. We will expose why we think that some areas of the map are not or painfully accessible.

At the end of this article, we will explain different techniques we propose to combine a highly iterative process with a required level of ceremony.

Dynamic aspect: Sequential or Iterative process

One of the most important decision when we have to choose a development process to start a project is the ordering of the activities: specification, design, code production, validation, …

The two main models are:

- The pure sequential model or waterfall model: Following this model we start with the complete definition of the requirements, following by the design of the solution, the development and finally the integration and validation phases. This model is often associated with the Big Design Up Front (BDUF) where the program's design should be completed (and reviewed) before the implementation is started.
- The iterative or incremental model (sometimes called spiral model): following this model, we deliver progressively the system based on increments (staging development); the result of each increment is a fully integrated subset of the features of the complete system (executable result).

It is well known today that, although the pure sequential model seems to be the most understandable and the most rational way to achieve the objective, in practice this model frequently fails.

The main advantages of iterative development are:

- The improvement of visibility on the progress, the quality, …
- The feedback at the end of iteration; having a better visibility, we can learn during the project and refine the development process from one iteration to another
- The response to change coming from customer (changes in requirements), from organization (changes in company strategy), from technical environment.
- The ability to manage the risk

Agile or less agile methods may get those advantages when practicing iterative development.

Ceremony aspect

A. Cockurn [Cockburn] defines the ceremony level as: "*The amount of precision and the tightness of tolerance in the methodology*".

Ceremony is also generally considered as implying much documentation and meetings.

In ceremony-based processes, requirements (system and software), architecture, design, … are extensively documented and controlled (for example with extensive traceability mechanisms),

It has for long be considered as a good way to control software development and evolution, but the practice shown that this did not imply high quality software, or client satisfaction, especially for big and long projects.

As of today, agile process developments propose to re-center on the software rather than on the documentation, and in some cases advocate that "the only deliverable is the code" or more appropriately "just enough" ceremony.

Obviously, ceremony is still needed in some cases, for example when required by the clients (but those clients also begin changing their minds in the agile way), or to obtain certifications, or for distributing information in large and distributed teams,…

Process dynamic aspect combined with ceremony level: the *Process Map*

The two previous aspects: the dynamic (iterative) aspect of a process and the ceremony level are crucial parameters for the choice or customization of the development process. Per Kroll [Kroll] and Craig Larman [Larman]) classify several development processes based on those .two axis, on a so called *Process Map*.

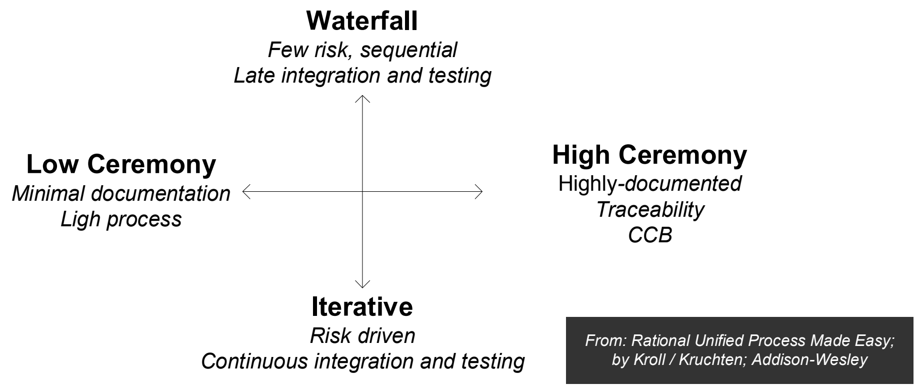

Figure 1: Process Map

What is the meaning of each axis? Although there is no formula defining the level of ceremony and the dynamic aspect, we can suggest several metrics.

First, we can determine the dynamic aspect:

- By the duration of an iteration: For a one-year duration project, one week length seems to be very iterative, one month seems to be the average of iterative process, or
- By the number of iterations: a process with one iteration is a pure waterfall…

The ceremony level is more difficult to quantify because it depends on the different practices and how the communication way (e.g. review report, minutes of meeting, …). Based on the "Lean" approach, we can estimate the level of ceremony by the percentage of *waste* effort from client perspective due to the ceremony.

Taking the example of requirement capture and validation activities, we may have two very different approaches:

- Using high ceremony process: the requirements are captured by writing a specification document. A validation test scenario document is derived from the first document and at the end of the development, validation test reports can be created to communicate which tests success and which ones fail.

  Above this first schema, we can add traceability, review reports, change control mechanism, …
- Another approach with very low ceremony level can be: the requirements are captured via a Wiki and a tool like FITnes [FITNES] based on the Wiki information (input and expected values) validates continuously which requirement is correctly implemented.

  Although this approach is not always fully applicable and has some weaknesses, we have a direct link between the requirements and the validation tests. The level of ceremony is minimized.

The following figure suggests a scale for the two axis.

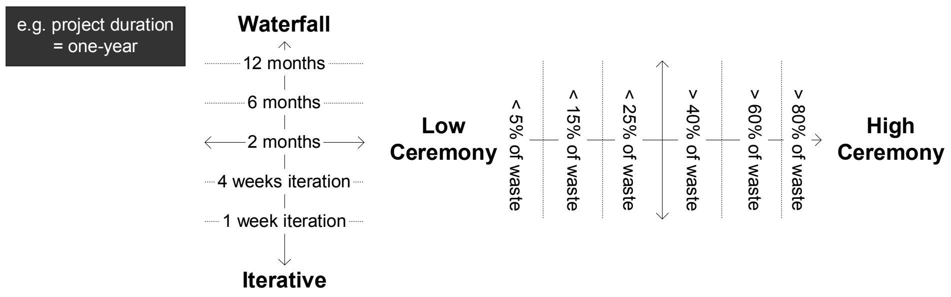

Figure 2: Process Map with scale

To illustrate the *Process Map,* let us start with a simple example of a common practice called *retrospective* or *post-mortem* analysis.

The goal of a *retrospective* is to gather information about the past, to inspect and adapt the methods and teamwork. You can find more information about (agile) retrospectives in [Derby].

In agile projects, it is generally a rather informal meeting (timebox meeting), wherein all the team members participate, and is made to get the lessons learned, and the points to be improved for the next iteration. Obviously, an iterative process will particularly show its strength against a pure waterfall one, if people are given the opportunity to detect problems, and to correct them rapidly and regularly.

In more ceremony-oriented projects, retrospectives focus on the retrospective document, documents, templates, metrics, … In a pure waterfall life-cycle, the retrospective occurs at the end of the project (to give feedback for future project);

The next figure shows how different project types can implement this same retrospective practice.

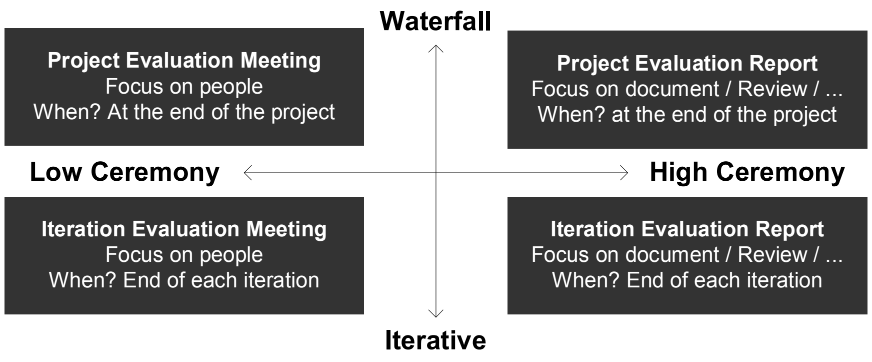
Waterfall
Project Evaluation Meeting
Focus on people
When? At the end of the project
Project Evaluation Report
Focus on document / Review / ...
When? at the end of the project
Low Ceremony
High Ceremony
Iteration Evaluation Meeting
Focus on people
When? End of each iteration
Iteration Evaluation Report
Focus on document / Review / ...
When? End of each iteration
Iterative

Figure 3: How to implement *Retrospective* practice

Development processes are made of many such practices, but generally speaking these practices have a similar level of ceremony/dynamic, and are thus coherently placed on the process map, as shown in the following paragraph.

Process classification

Craig Larman (see [Larman]) proposes for example the following classification of several well-known processes.

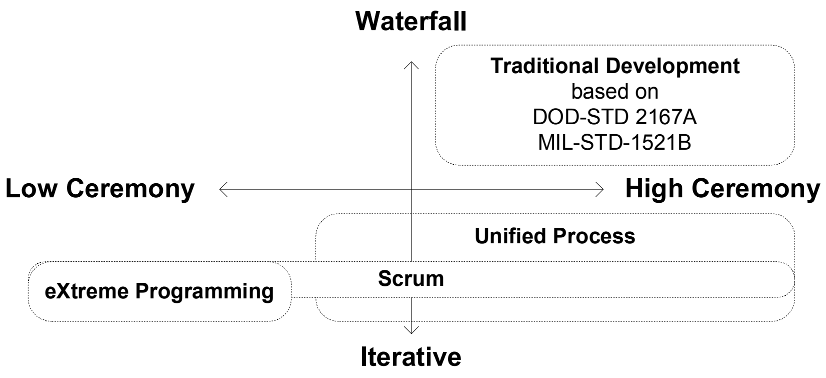

Figure 4: Major processes

Each process is presented as an area in the *Process Map*, this area is the allowed customizations of the process for one particular instance.

Death zones

Are the four quadrants achievable? We do not think so…

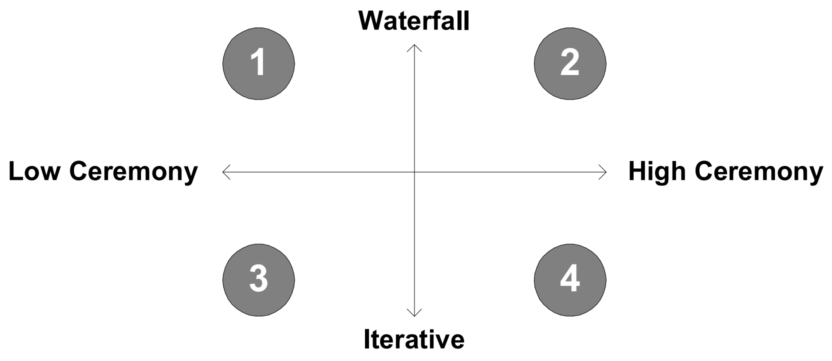

Figure 5: The four quadrants

1. A pure or quasi-sequential process with a very low ceremony seems to be achievable only for extremely *simple* projects: short duration (less than 1 month), very small team (1 or 2 people) and with a low level of risk.

   For normal projects such an approach without any possible control, visibility, and correction possibilities seem to be perilous or even suicidal. Hopefully, this quadrant is very uncommon.

2. The second quadrant is the area of pure or quasi-sequential life-cycles with high ceremony levels. This process follows the analogy of a build chain process (for example car assembly lines). In theory this process family seems attractive and pleasant from a management point of view, because they seem to be well defined and controlled; in practice unfortunately those processes have the highest level of failures, especially for complex projects.

3. Highly iterative and with a minimum of ceremony are the characteristics of agile processes as XP, DSDM, Crystal, … which already shown successes in small and middle scale projects, and begin to be considered for larger scale projects (see [Eckstein] and [Leffingwell]).

4. Iterative process combined with a high level of ceremony is the recommended instantiation of the RUP for large scale or critical project. But is it achievable in practice? Is it feasible to have for example four weeks iterations with a very large number of documents, traceability, formal reviews, …?

   In practice, we think that those projects have to increase the length of the iteration (3-6 months) or reduce the ceremony effort (e.g. documentation is no more updated, …) to be practical.

The practicable diagonal law

The difficulty to have a highly iterative with high ceremony is the weight of ceremony realization and maintenance during the execution of an iteration. Each document, each review, each traceability information has to be realized and maintained during the execution of any iteration.

As such, the ceremony is sometimes seen as a load that we have to push ahead simultaneously as we progress:

Photo by Dey
(http://www.flickr.com/photos/dey/98476303/)

This load has to be reasonable compared to the dynamic and complexity level of the workload.

As such, we can conclude that the workable processes, from the practitioner point of view, are located near the diagonal:

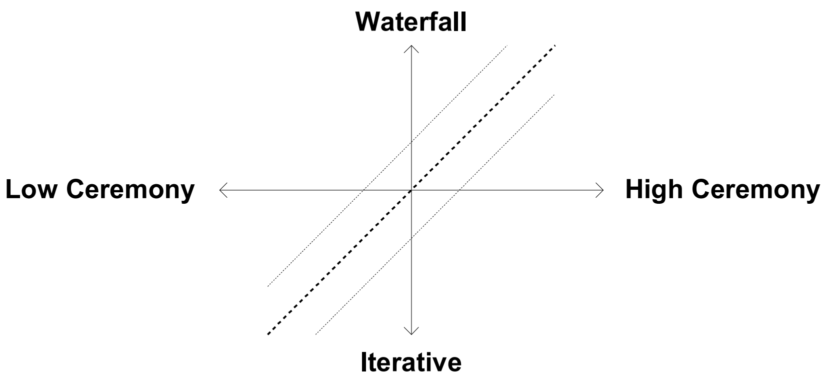

Figure 6: Practicable diagonal law

When ceremony is required…

As already explained, the ceremony is sometimes required directly by the client, by a regulation office or internally to communicate between large or distributed teams.

Following our previous reasoning, is an iterative project with the required level of ceremony reconcilable? And how?

The two straightforward solutions, represented on the following process map, are:

(a) Reduce the number of iterations and go to a more waterfall process. This solution has shown its limitations in the past.

(b) Reduce the level of ceremony to the minimum required for the certification. We can reduce this cost by applying the "**D**on't **R**epeat **Y**ourself" principle, by focusing on the essential..

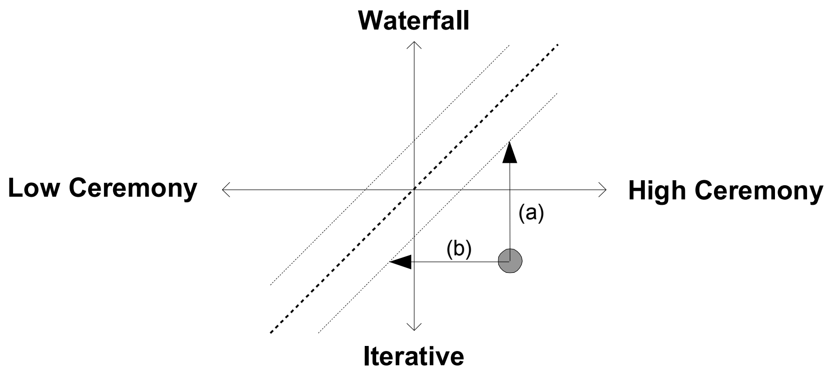

Figure 7: straightforward solutions

Another solution (c) is to increase the area of practicability by reducing the cost (effort and time) of the ceremony realization and maintenance during the life-cycle, using for example helper tools (e.g. Literate programming, CASE tools to obtain the design models of the application, quality metrics tools to document the quality, …).

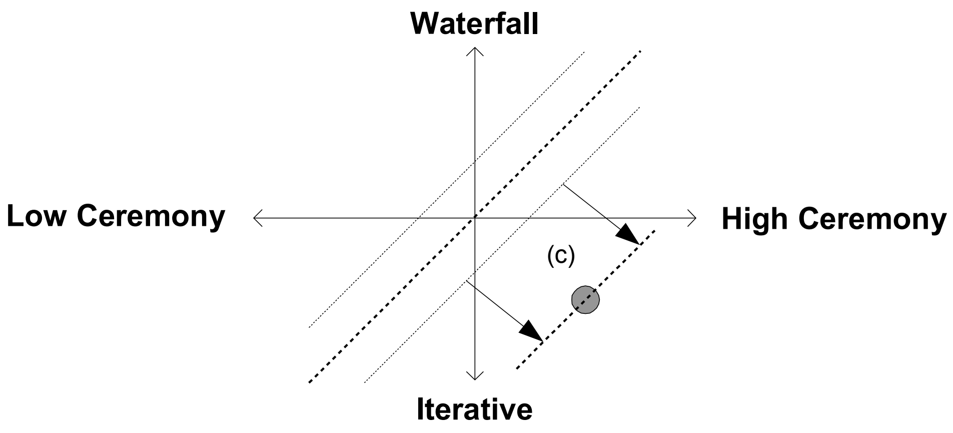

Figure 8: Increase the area of feasibility

None of these solutions is really satisfactory. But…these were static views of the process, as the *Process Map* gives a static classification of the process. Now, *what if we change the level of ceremony from one iteration to another?*

For example, we may have the first iterations of the project having a low ceremony level, then followed by a high ceremony iteration. This approach is illustrated in the next figure.

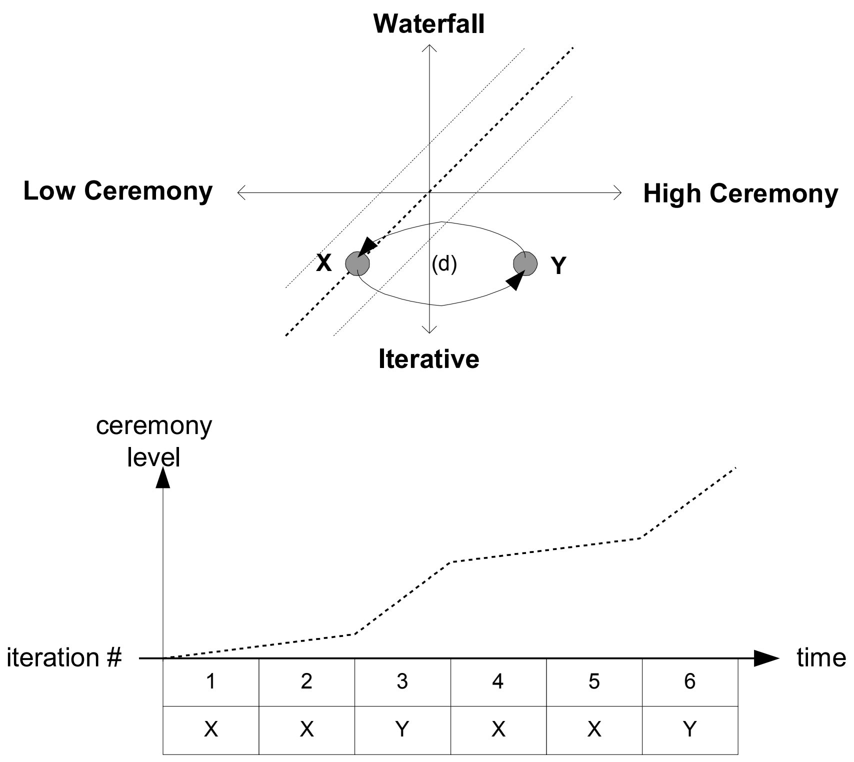

Figure 9: Different levels of ceremony per iteration

Such a solution allows developing the first iterations rather freely (iteration 'X'), and introduces ceremony only when the application stabilizes itself (iteration 'Y'), to document the real implemented solutions, without having to maintain the documentation in the early discovery phases.

In most agile developments, we have the iteration rhythm and the release rhythm (several iterations). The last iteration of a release is the natural choice to increase the level of ceremony.

Based on Unified Process family, we can use the internal "*Lifecycle Architecture*" milestone (i.e. the end of Elaboration phase) to increase the level of ceremony to the "just enough" level.

Obviously, we may combine the three last explained methods: (b), (c) and (d) because we may keep the same dynamic..

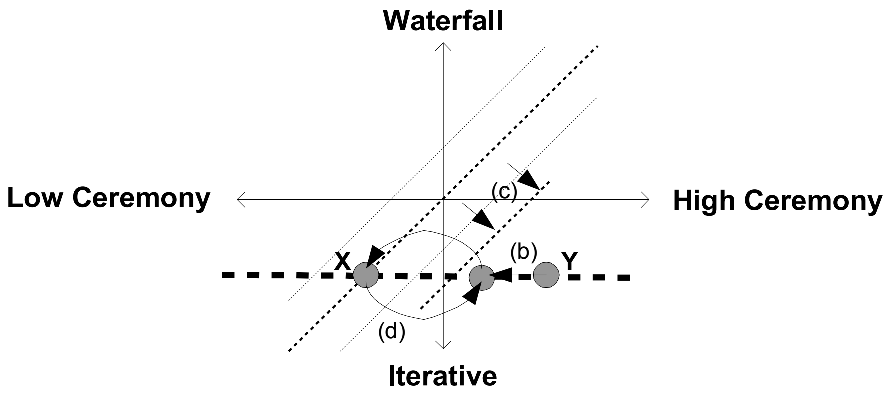

Figure 10: Combined approach

Conclusion

We think that tailoring the ceremony per iteration is really the solution that should be proposed to combine agile and dynamic processes with required ceremony levels.

The customization of such a process is naturally dependent on the project itself, on the level of ceremony required, and on the freedom the team has on its process.

As conclusion, we have the following recommendations for people thinking of agile and dynamic processes in ceremony-requiring environments: when you are required to attain a specific level of ceremony on a project:

- Apply the lean vision and DRY principle: examine the utility of each artifact, practice, …., and keep only the one really useful and needed.
- Reduce the cost of ceremony by automating the realization and maintenance of ceremony (limited solution: tools are not “silver bullets”).
- Finally, use the (d) approach, to limit the ceremony constraint at the stages where the software stabilizes itself, and where the documentation will not have to be updated many times. (difference between iterations and release).

Acknowledgments

Many thanks to our friends Antoine CHOPPIN and François ROUCOUX for their valuable comments.

References

[Derby]	Agile Retrospectives: Making Good Teams Great; Esther Derby, Diana Larsen; the Pragmatic Bookshelf; 0-9776166-4-9
[Eckstein]	Agile Software Development in the Large: Diving Into the Deep; Jutta Eckstein; Dorset House Publishing Company; 0932633579
[Kroll]	The Rational Unified Process Made Easy; P. Kroll, P. Kruchten; Addison Wesley; 0-321-16609-4
[Larman]	Agile and Iterative Development: A Manager's Guide; Craig Larman; Addison Wesley; 0-13-111155-8
[Leffingwell]	Scaling Software Agility: Best Practices for Large Enterprises; Dean Leffingwell; Addison Wesley; 0-321-45819-2
[Poppendieck]	Lean Software Development: An Agile Toolkit for Software Development Managers; Mary Poppendieck, Tom Poppendieck; 978-0321150783