How to Reduce Lead-Times in long Devops Agile Projects

Mahmoud Abou-Gabal
Ottawa, Canada
mahlink03@gmail.com

***Abstract*: Longer Lead times in Devops process has shown to be a challenge to manage and costly for organizations if unplanned for long-term (longer than 1 month) agile projects. According to [11], typical agile lead duration varies between from one to four weeks. Therefore, agile by design is supposed to be short. However, if agile exceeds the threshold of more than one month, then organization needs to introduce ongoing/controlling mechanism to bring it back to be short again. The aim of this paper is to outline how such ongoing planning/controlling mechanism can be developed for long-term agile projects. In recent years, almost 50% of organization has suffered economical losses [12] and teams burn-out due long-term agile projects. Therefore, it is key to understand the core agile process for Digital Transformation; Continuous Integration (CI) and Continuous Deployment (CD) to help manage the key challenges which this paper outlines in three sub-sections. Also, this paper demonstrates how innovation process in agile should help organization to excel as successful organizations like in [3] have done.**

Keywords*— *Agile, Project Management, Digital Transformation (DT), Devops, Cloud deployments, Continuous Integration (CI) and Continuous Deployment (CD)

I. Introduction

Agile management is used today in software development and in Digital Transformation projects. Agile was originally formed by 17 technical experts in software development back in 2000 [16,7]. They recognized two key opportunities that achieving this agile goal would make possible:

1. Shortening the delay of benefits to users to resolve the product-market fit and development graveyard problems
2. Getting feedback from users quickly to confirm the usefulness of new software and continue to improve on it accordingly.

It clear that from design of agile cycle that shortening the cycle has a lot of benefits, but over the years it showed us agile can fail [1] in many organizations that use it without realizing agile intent of being short cycle. And for this reason today, some organizations are suffering to manage long lead-times agile projects and have high failure rates of completion (more than 47%) [12]. This paper recommends to include ongoing planning/controlling mechanisms similar to waterfall but reduced to fit agile methodologies. Next sections of the paper look at how organizations can better manage time and how long agile cycle projects can adopt some tools from waterfall for better planning (section II), also CI/CD process breakdown is considered to better manage the core elements of software development (section III), and how innovation process is needed to manage better team's burnouts which in return produce shorter lead-times.

II. Time Management

As shown in figure 1, waterfall is known for having long planning cycle and could be inflexible for scope changes later in the execution phase of the project. On the other hand, agile is short therefore no long-planning cycle is required, and changes can be made easier. However, when agile projects face continuous scope changes in organization today, in which extends the cycle from short to longer cycles, longer agile projects are at risk of being completed on time which leads to longer lead-times. Therefore, for these longer agile cycle projects, ongoing-planning and controlling mechanisms are needed.
Typical agile cycle tends to be between 1 week to 4 weeks (max) as per [11], this is the time threshold that management needs to consider for short agile. Longer than one month management should start questioning their agile process and should consider other options how to better manage longer agile projects.
In this paper suggest including ongoing planning/controlling mechanisms (adopted from waterfall) for longer agile projects. As shown in figure 2, ongoing planning has a longer planned cycle than agile but shorter than waterfall (which is in between). This planning/controlling mechanism should bring

back the agile to be shorter cycle and manage scope changes better.

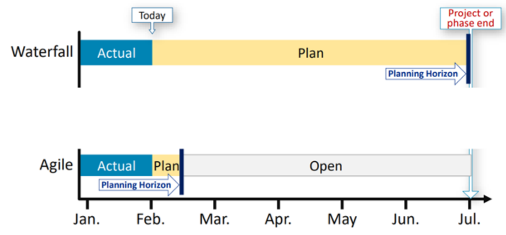


Figure 1: Planning and Controlling comparisons between Waterfall and Agile

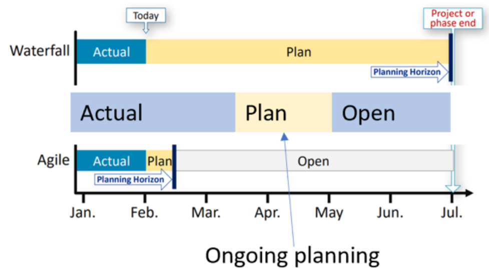
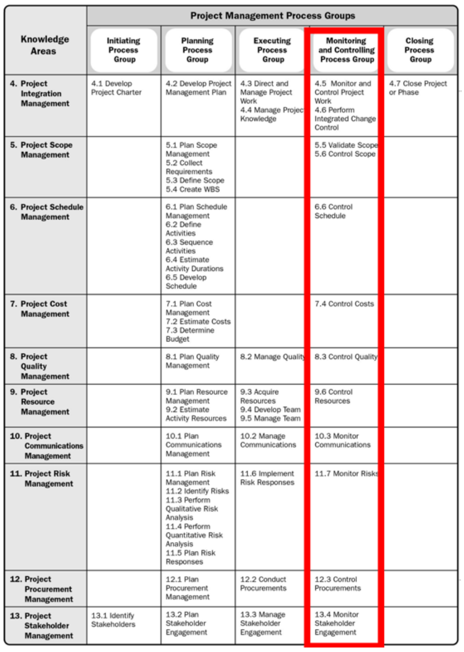


Figure 2: Ongoing Planning/Controlling for better agile that exceeds shorter cycles (more than a month)

Some of the tools that should be used in the ongoing planning are shown in Table 1 [8] from the waterfall approach. This is the knowledge table where the controlling mechanism, like monitor project integration, perform change control, validate the scope, control the schedule with key deliverables, control costs, and quality.

Table 1: Project Management Knowledge Table 1 (from waterfall)



Knowledge Areas	Project Management Process Groups				
	Initiating Process Group	Planning Process Group	Executing Process Group	Monitoring and Controlling Process Group	Closing Process Group
4. Project Integration Management	4.1 Develop Project Charter	4.2 Develop Project Management Plan	4.3 Direct and Manage Project Work 4.4 Manage Project Knowledge	4.5 Monitor and Control Project Work 4.6 Perform Integrated Change Control	4.7 Close Project or Phase
5. Project Scope Management		5.1 Plan Scope Management 5.2 Collect Requirements 5.3 Define Scope 5.4 Create WBS		5.5 Validate Scope 5.6 Control Scope	
6. Project Schedule Management		6.1 Plan Schedule Management 6.2 Define Activities 6.3 Sequence Activities 6.4 Estimate Activity Durations 6.5 Develop Schedule		6.6 Control Schedule	
7. Project Cost Management		7.1 Plan Cost Management 7.2 Estimate Costs 7.3 Determine Budget		7.4 Control Costs	
8. Project Quality Management		8.1 Plan Quality Management	8.2 Manage Quality	8.3 Control Quality	
9. Project Resource Management		9.1 Plan Resource Management 9.2 Estimate Activity Resources	9.3 Acquire Resources 9.4 Develop Team 9.5 Manage Team	9.6 Control Resources	
10. Project Communications Management		10.1 Plan Communications Management	10.2 Manage Communications	10.3 Monitor Communications	
11. Project Risk Management		11.1 Plan Risk Management 11.2 Identify Risks 11.3 Perform Qualitative Risk Analysis 11.4 Perform Quantitative Risk Analysis 11.5 Plan Risk Responses	11.6 Implement Risk Responses	11.7 Monitor Risks	
12. Project Procurement Management		12.1 Plan Procurement Management	12.2 Conduct Procurements	12.3 Control Procurements	
13. Project Stakeholder Management	13.1 Identify Stakeholders	13.2 Plan Stakeholder Engagement	13.3 Manage Stakeholder Engagement	13.4 Monitor Stakeholder Engagement	

As shown in figure 3a, short -term agile doesn't require ongoing planning. However, for longer cycle agile as shown in figure 3b, ongoing planning and monitoring procedures are needed to be integrated in agile process.

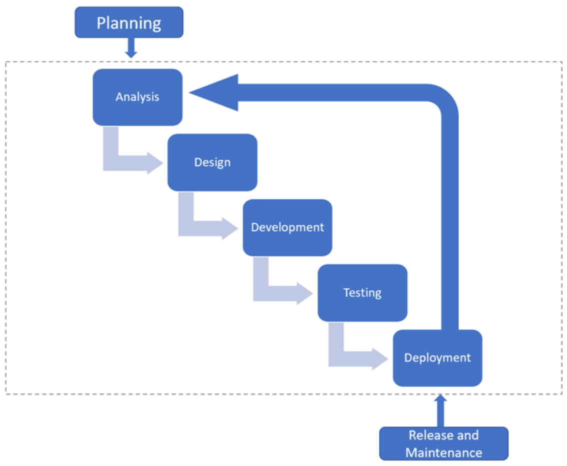


Figure 3a: Short agile

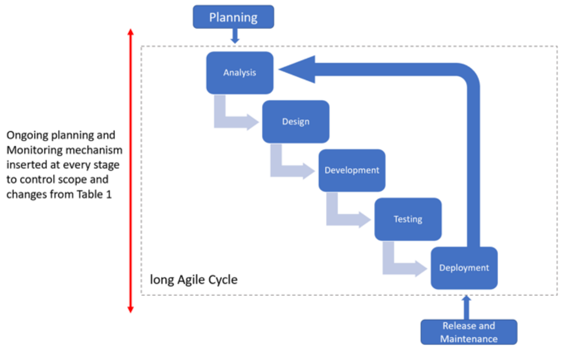

Planning
Analysis
Design
Development
Testing
Deployment
Release and Maintenance
Ongoing planning and Monitoring mechanism inserted at every stage to control scope and changes from Table 1
long Agile Cycle

Figure 3b: Long agile

Equipped with Table 1, it is important also to understand the key or core process of agile process for digital transformation projects, which is the Continuous Integration (CI) and Continuous Deployment (CD) of software solutions. This paper looks at three important aspects of the CI/CD process and following subsections explain the key elements of CI/CD process for management to plan/control around them:

1. Overall technical aspect of CI/CD process.
2. Agile Team + Organization culture.
3. Overall agile + Organization toward strategic fitting

III. CI/CD PROCESS

As per [1], CI/CD is a method to frequently deliver apps to customers by introducing automation into the stages of app development. The main concepts attributed to CI/CD are continuous integration, continuous delivery, and continuous deployment. CI/CD is a solution to the problems integrating new code can cause for development and operations (Devops) teams.
Specifically, CI/CD introduces ongoing automation and continuous monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. Next subsection explains some of the CI/CD process and how management can manage around it.

A. *CI/CD Process*

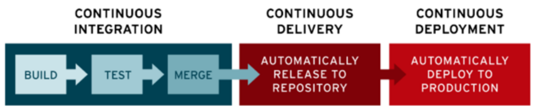


Figure 4: CI/CD process/pipeline [1]

For Digital Transformation to happen, CD/CI process sets in the center of this crucial transformation. That is why understanding the overall process/pipeline will help manage the agile Devops teams.

As shown in figure 4, the overall process [1], shows that continuous integration, continuous delivery, and continuous deployments are the three core steps of this transformation. In [2, 15], looking a bit deeper on the technical side, shows the seven important steps in CI/CD;

1. The trigger: The best pipelines are triggered automatically when new code is committed to the repository.
2. Code checkout
3. Compile the code
4. Run unit tests
5. Package the code
6. Run acceptance tests
7. Delivery or Deployment

Steps 6 and 7 are usually the longest steps [13] for some organization due to manual work and lack of automation cultures, this will increase the lead-times which a lot organization is facing today that has economic and high burn out levels on agile teams. Therefore, it is important for organization to first develop a good culture to aid agile teams to be more productive in CI/CD process.

B. *Agile team + Organization Culture*

An important thing to remember is that team needs to have in mind that the end goal of what is important for the organization, and everyone is in it to make it happen. This is where team's silos or boundary set for the roles (Developer vs tester vs security vs operations) needs to be removed. It may mean in Devops world that everyone would be working on cross boundaries to quickly deliver the end goal. As other competitor knows this, organization would be at risk of the agile team think in silo roles which will increase the handover times between teams, Developer develops, takes over Quality Assurance (QA) may finds bugs or not, handover to Security may see issues or not, then passes over to operations, operations environments does take the developer new code, then back again to this handover mode, Ops→ Sec→ QA→ Developer. So basically throwing over fence type mode will increase Leadtime especially if no automation is present.

Organization needs to break this mindset/culture of silos between teams therefore reducing handover time by removing boundaries between roles. Everyone on the team should be involved at same time to deploy the new code. Organization's today needs shifting responsibilities across the roles in Devops agile teams as per [4]. There are competitors and organizations who applies successfully agile management and CI/CD as shown in [3]. This is done by combining all the building blocks of CI/CD as shown in next subsection.

C. *Overall Agile + Orgnanization fitting for strategic end goals*

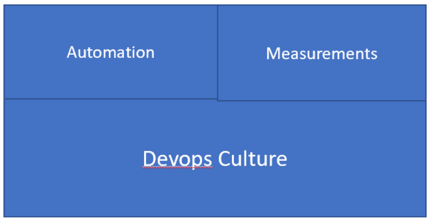


Figure 5: building block for good CI/CD process

Figure 5 shows, once Devops team has the right culture (silos are removed) which is the base, then automation and measurements can be defined later in the CI/CD process to make it more productive and faster to deploy agile projects. In this paper we touch slightly on both key measurements and automation to help guide the management within the organizations.

For measurements, general key metrics management can define as in [9] are the following.

1. **Deployment frequency:** How often code is deployed to production.
2. **Lead time:** How long it takes from code being committed, to code successfully running in production.
3. **Mean time to repair (MTTR):** How long it takes to recover normal service when an incident occurs.
4. **Change fail rate:** The percentage of changes to production that result in poor service and require remediation.

If management had to choose one basic metric to follow for Devops and agile team, would be lead time. Lead time measure is important as it proves effectiveness of the organization to quickly adopt to changes.

For automation, as discussed in sub section A, that testing, and deployment of CD/CI process needs automation as more manual work could be dragging the team and make thing inefficient and long to execute. As shown in [10], automation helps to; 1. Eliminate performance bottlenecks, 2. Minimize communication gaps between the development, operations, and quality assurance teams, 3. Introduce mechanisms that facilitate agility through standardized processes. For organizations to excel at agile process, which also goes hands in hands with automation, is adding innovation cycle in the agile process execution shown in the next section. Automation is part of innovation because it needs time that organization will see the benefits as was done by successful organizations in [3].

IV. Innovation Cycle in the Agile and How Successful Organizations Has Done It

As shown in figure 3a, it shows how team can go in the never-ending cycle (Analysis-Design-Development-Testing-Deploy) that can get very stressful if repeated for many releases on the same team again and again. According to [14], agile has experienced burn-out and fatigue due being on continuous agile cycles. This paper purposes an innovation process after every software release completed as shown in table 1. So, by breaking the teams by A, B and C (as an example), and rotate them every release so every team goes out and into the innovation cycle. This way team can take a break from the never-ending cycle which creates burnout and stress therefore compromising the software releases for successful digital transformation. To manage the innovation cycle itself, there are many good literatures like in [5]. Esty from [6] shows good example of agile team in action using innovation where they have for their agile teams a specific blog on how they innovated for the public to see and learn from.

Also, in this paper shows how innovation can be inserted as shown in figure 6. Whereby every release some questions are asked if agile team was part of long/short agile cycle and for how many times. For successful automation teams needs break and this is where in innovation cycle can bring new ways for automation as done by Esty in [6]. Innovation is a stress relief for the teams as shown in [5].

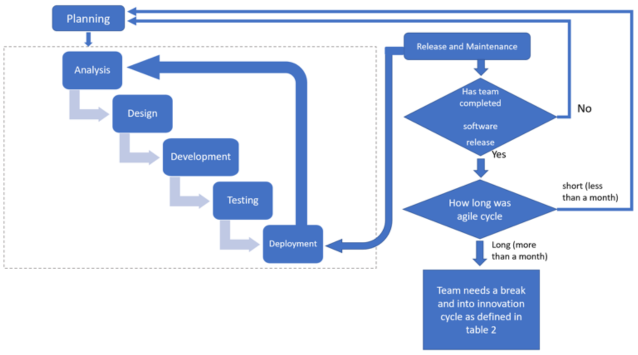

Planning
Analysis
Design
Development
Testing
Deployment
Release and Maintenance
Has team completed software release
No
Yes
How long was agile cycle
short (less than a month)
Long (more than a month)
Team needs a break and into innovation cycle as defined in table 2

Figure 6: Modified Agile Process to include innovation process

Table 2: Innovation cycle inserted in Agile

Software Releases	Release 1	Release 2	Release 3
Team A	In Agile cycle	**Innovation cycle**	Support B and C
Team B	Support Team A and C	In Agile Cycle	**Innovation Cycle**
Team C	**Initial Exploring/Innovation**	Support B and A	In Agile cycle

V. Conclusion

Agile by nature is short cycle and, in this paper, shows how to introduce ongoing planning for long agile projects cycle to better manage agile teams for shorter lead-times. CI/CD is key process for the software development which core of DT projects. It was shown by understanding the CI/CD process and establishing good agile team culture (removing silos) can build better base for measurements and automation. Innovation process is also proposed to help manage team's burnout and reduce stress level. Automation needs time and with innovation process, better opportunities is given to the team to better innovate.

REFERENCES

[1]. What is a CI/CD pipeline?, https://www.redhat.com/en/topics/devops/what-cicd-pipeline

[2]. "The 7 essential stages of a CI/CD pipeline", https://www.tutorialworks.com/cicd-pipeline-stages/

[3]. Companies killing it in Devops, https://techbeacon.com/app-dev-testing/10-companies-killing-it-devops?amp

[4]. Devops mindset, https://devops.com/the-evolving-devops-journey-continuous-mindset-starts-with-cultural-change/

[5]. Digital Transformation: Practical Approach to Manage Transformation (The Change) Successfully within the Organization, https://ieeexplore.ieee.org/document/9488603

[6]. Esty Blog: Etsy Engineering | Code as Craft

[7]. Agile Methodology, https://www.planview.com/resources/guide/agile-methodologies-a-beginners-guide/history-of-agile/#:~:text=It%20all%20started%20in%20the,new%20software%20to%20market%20faster.

[8]. Knowledge Table from PMI, https://www.project-management-prepcast.com/pmbok-knowledge-areas-and-pmi-process-groups

[9]. DevOps metrics for success, https://www.dynatrace.com/news/blog/devops-metrics-for-success/

[10]. Why & How To Automate DevOps Practices, https://www.bmc.com/blogs/automation-in-devops/

[11]. Agile Scrum Sprint Length: What's Right for You? https://www.methodsandtools.com/archive/scrumspringlength.php

[12]. Why do 47% of Agile Transformations Fail? https://www.scruminc.com/why-47-of-agile-transformations-fail/

[13]. CI/CD Pipeline, https://semaphoreci.com/blog/cicd-pipeline

[14]. How to spot Agile burnout https://www.wrike.com/agile-guide/faq/what-is-agile-burnout/

[15]. *S. Artelt, "*Analysing the Impact of Agile Project Management on Organisations*", 2021 IEEE European Technology and Engineering Management Summit (E-TEMS), September 2021.*

[16]. R. Hoda, N. Salleh and J. Grundy, "The rise and evolution of agile software development", IEEE Software, vol. 35, no. 5, pp. 58-63, September 2018.