

## **Extreme Programming for Execution of Large and Geographical Distributed Software Projects**

**Mrs. Nagalambika<sup>1\*</sup>**

Research Scholar, Department of MCA,  
Dr. Ambedkar Institute of Technology,  
Bangalore, Karnataka, India<sup>1</sup>.

\*E-mail: nagalambika.swamy@gmail.com  
manjuarjun2004@yahoo.com

**Dr. L. Manjunath Rao<sup>2</sup>**

Professor and Head, Department of MCA  
Dr. Ambedkar Institute of Technology,  
Bangalore, Karnataka, India<sup>2</sup>.

E-mail:

### **Abstract:**

There have been a number of software development processes. Based on their advantages and disadvantages, they are accepted and utilized. But none of the development process can be claimed to be suitable to all the projects as the development context/scenarios differ. Therefore, there is a need to model the software development process based on the on-going situations. In this paper, we define and develop a software development process that uses Micro Service Architecture and DevOp culture for Extreme Programming (XP). The proposed software development process can be used for large, complex and geographically distributed software system. A literature review was conducted to understand the currently used software development process and architectures in extreme programming. Nevertheless, in practice, the usage of extreme programming in large scale companies/systems is not known. This study aims to evaluate the impact of the usage of XP process on the development of large-scale distributed systems, while taking online shopping services as a case study. A case study is conducted in an organization that develops software that are large-size and complex by modifying the extreme programming. Questionnaires were prepared and qualitative analysis was carried out. The case study aided in learning about the effectiveness of combining XP with DevOp. Further, outcomes of Crystal-Clear Methodology that depends on people rather than processes and Extreme Programming were compared. As a result of this process, XP can handle large, complex and geographical distributed software systems. Developed software becomes much faster, cost effective, loosely coupled, deployable across the globe. It is observed that practices of Extreme Programming when adapted in the project gives rise to output like people factor and also helps creating ideas and solutions for complex design issues. This is the approach for conceptualization and implementation of overall systems.

**Keywords:** Extreme Programming; Large and Geographically Distributed Software projects; Agile Software Development; Microservice architecture; DevOp.

### **1. Introduction**

Architecture is an important component of a software system [1]. A microservice architecture consists of a group of tiny and autonomous services. The services offered are self-contained and should be implemented in a single business capability. These microservices are small, independent and loosely coupled. These services are coded and maintained by a one compact developer team. Services can be deployed independently. The existing services can be updated without the need of reconstructing and readjusting services in whole application. DevOps are set of practices followed in software development and IT operations. Development components are Build, Code, Test and Plan whereas Operations components are Monitor, Display, Operate and Release. Key process of Devops is continuous development and integration where the development and coding of the software is taking place [2].

The traditional models like waterfall, Rational Unified Process (RUP), spiral and incremental models do not suffice the current software functionality requirements because of which popularity of Agile methods have increased [3, 4, 6]. It is observed that Carnegie Mellon University's Software Engineering Institute (SEI) has developed a series of software technology with architecture-centric methods that are used for architecture design and analysis [9]. Such methods have been attempted to fit into software development processes that are in demand. In concern to the methodologies used in traditional software development, all the necessary requirements and specifications of software have to be collected when the project starts and these don't change later. But this is not the case. To incorporate the changes in the development process, we detail the agile methods that are flexible and people-oriented. In agile, the change of software requirements can be incorporated effortlessly in the design at any stage. This makes agile methods flexible. Hence, the agility process gives faster software delivery and also creates instant business values to the customers.

Agile methodologies came into existence from 1960's, yet they were regularized by signing an agile manifesto [3] in 2001. The agile methods such as Extreme programming, Feature-Driven Development, Crystal Clear and Scrum [8] are in accordance with the agile manifesto. Agile methodology is a light-weight development process, human-oriented and adaptive. The changes in the software design can be easily incorporated. Agile process is used to create adaptive and resilience architecture. Simple design, iterative coding, better design practices, delay in taking decisions and flexible plan are the strategies used by software architecture.

At present, the data that is processed is huge and with the trend of big data, it is even more essential that the architectures used in development process should be capable of handling and processing such data. The big data projects concentrate on requirement analysis of users and the architectures used for big data are standardized and changes in that rarely happen (Agile big data analytics). Microservices solves architectural bottlenecks whereas Agile solves Engineering bottleneck.

Extreme programming is one methodology that is extensively used in agile development methodologies [7, 10]. It was invented back in 1999 by Kent Beck and that was later refined in 2004. The XP's first edition began with four values, fifteen principles, and twelve practices [13]. The first one and the new version differ considerably. Later, after five years of its first edition, XP has been critically reviewed and tuned so that it incorporated the recent development in industry. The latest XP has thirteen primary

practices and eleven corollary practices. As these are not fundamental practices it's difficult to implement them before training the people.

In this paper, we explore the association and interlink age between Extreme Programming framework, architecture-centric design and methods of analysis. In this research, XP is selected as it is one of the renowned agile practices. In the XP development model, iterations are used for development and system requirements are recorded by the customer by means of user stories. In planning stage, the developers along with customer determine the required functionality to be developed. The test situations are extracted from user stories with interactions the customer and development team. Later, the programming interface is designed by the developers. Codes are written in such a way that these designs should to match testing requirements. Further, the design is refined for matching the code's needs. Finally, the output of the design is a product for the customer. But the quality of the product depends on the expertise of the members in development team. The activities can be informed and the development process can be formalized by emphasizing quality attributes that could be taken into considerations for architectural decisions.

### **“The Quality Attribute Workshop” (QAW)**

In an XP project, the requirements are collected, captured, documented, and analyzed by a system analyst. At the beginning of every iteration, user stories need to be evaluated and system analysts lead and coordinate for modeling the system. The outcome is a detailed specification for the system's functionality. The QAW is often conducted in the initial development process to show specification of the quality attributes in the form of scenarios. The stakeholders are involved from the beginning of the life cycle to fulfill quality attribute requirements and therefore it is stakeholder-centric [8].

### **Early design: The ADD method**

The system is developed incrementally and hence it is refactored whenever new functionalities are not supported. The initial iteration is very important when defining the overall structure of the system. The ADD method highlights on the overall system structure which often is ignored by XP developers. This should be focused in initial iteration till final iterations, so that software architecture incorporates any changes that are substantial. The information about the architecture, constraints and quality attributes are displayed on a notice board in office rooms to keep all of them updated.

Extreme programming is suitable for small sized projects with small teams working on them. It concentrates on customer integration, immense testing, development and documentation centered around code, paired programming and restructuring. XP is an agile software development methodology that is based on values like clarity, courage, reporting, presenting and feedback [15].

- Extreme Programming fits in for small projects and not appropriate for medium, large-scale and complex projects.
- If programmers are located geographically, XP is not the best option.
- Extreme Programming is a code centric and not a design centric development. The absence of the XP design concept makes handling large projects difficult.

There is a need to define a design centric development and Quality aspect of development, research to be undertaken to narrow down on the apt extreme programming that will yield maximum benefits and boost the project performance towards achieving the successful deliveries. The aim of this paper/study is to define and develop a Software Development Process (SDP) that uses agile methodology with Extreme programming for micro service architecture. It is known that extreme programming is code centric rather than design centric and it is location dependent. In this research/study we develop a SDP where extreme programming can be used for large, complex and geographically distributed network along with microservices. With this approach, development process becomes fast, cost efficient and loosely coupled. DevOps are used that shortens the development life cycle and provide continuous integration to deliver high quality of software, with the above objectives we frame the following research questions.

1. Can we develop a unique framework for microservices that uses extreme programming?
2. Can the extreme programming be used for and large size and distributed software projects?
3. In contrary to XP being code centric, can it be used for design centric development?
4. Can we go away with pair programming practice as it leads to more expenses?
5. Is performance of extreme programming better than Crystal-Clear Methodology?

The paper organization is as follows: Section 2 describes the Methods and Materials that includes Methodology, the related work, proposed model and case study. Section 3 presents the Results and Discussion and the Conclusion in Section 4.

## 2. Methods and Materials

### 2.1. Methodology

We conduct a qualitative study to address the research questions posed. The methodology used in this study is as shown in Fig. 1.

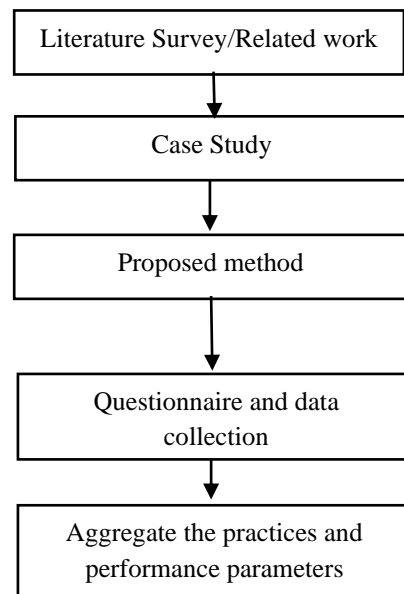


Figure 1: Study Methodology

First, to understand the cutting-edge information on ‘software development’, a literature survey was carried out. Keywords like software development process, agile methodology, extreme programming, DevOps and micro service architecture were used to search the research papers. The survey analyzed different processes that were and are in use for software development process. The literature survey focused on studies that implemented microservice architecture in extreme programming. Second, a case study was carried out on online order management system to identify agile practices. Such system was selected that develops the large size and complex software and use extreme programming. A questionnaire was prepared to access suitable for software development practices when the systems are large scale as well as complex [4]. Online open-source repository was used for this study, as companies can not disclose their data as it is confidentiality concerns and intellectual property of any organization.

## 2.2. Case Study: Online Order Management System

The case study was conducted on order management. First the microservices architecture is explored in this study. Microservices can be defined as tiny or small, self-sufficient and loosely coupled. Small development team manages each service and these services are a separate codebase and these services could be independently deployed. To update the current service, reconstructing and readjusting of the whole application is not required. Services are responsible for keep going their own information/data or external state unlike the traditional model, in which data persistence is handled by a separate data layer. Importantly, well-defined APIs are used by services communicate with each other. The implementation details of each of such services are masked from rest of services (Refer Fig.2). The key process of DevOps is continuous development and integration where the development and coding of the software is taking place (Refer Fig.3).

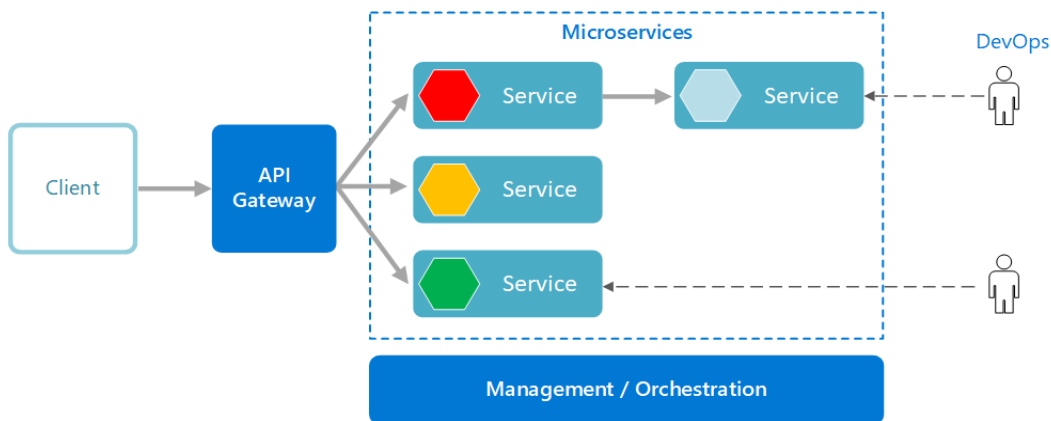


Figure 2: Microservice Architecture

Continuous integration stage most important stage of DevOps lifecycle. This software development practice requires developers make changes to the source code on a day-to-day or weekly basis. Then each

commit undergoes build, and in this process, problems are detected in early stage. Building code has the following procedure to follow:

Compilation → unit testing → integration testing → code review → packaging.

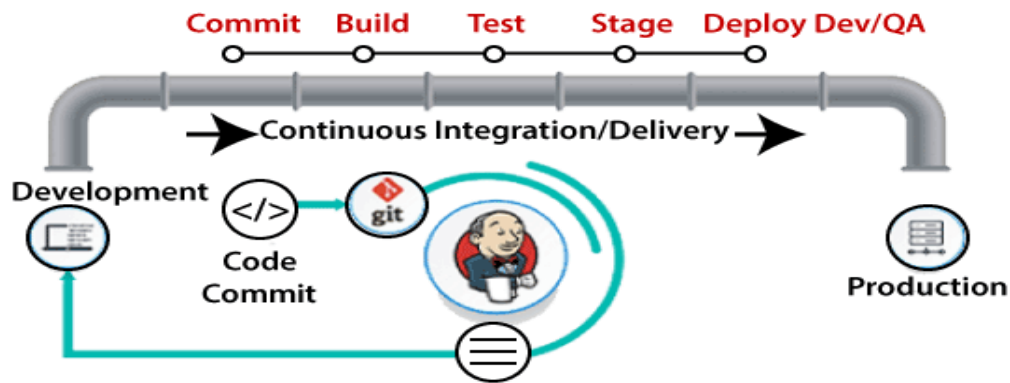


Figure 3: DevOps lifecycle

Agile and plan-driven approaches can be balanced by making use of their strengths and compensating for their weaknesses. Agile approach is appropriate to scenarios that have rapid changing requirements. Agile methods should be used in developing complex, large scale applications as more kinds of architectural information can be incorporated [12, 13]. The proof that architect role is recognized in zero-feature release, the architectural projection, and agile practices. According to Steve McConnell, “the evolutionary delivery life cycle is a best practice”. Evolutionary delivery is an architecture activity that balances both, control and flexibility. It’s movement in one direction depends how much change requests by customer are accommodated. A prominence on the system features directs towards evolutionary prototyping and on the system’s core architecture towards staged delivery. Alistair Cockburn identifies Extreme Programming (XP) appropriate for smaller projects with team members of 4-14 .It’s demonstrated that a solution can be obtained by transforming to XP for developing complex and large-scale applications by plan-driven methods. But these need architectural plans that are of high level, design patterns and architecture centric solutions rather than simple design. Restructuring needs to be considered.

Because of insufficient design and lack of documentation, XP cannot be used in projects. Yet, there are reports that show XP implemented to medium projects and as well as to large projects and complex software settings by modifying XP to suit them. In this case study, for a large-scale project "FinApp" an agile approach was used by reforming the existing practices of XP like layered approach instead of short cycles, design up front, customers influenced by business analysts, management of developers, reuse with restructuring, organizations/systems with flat hierarchies with controlled empowerment and partial pair programming, all of that chosen by developers. It was an experience where agile practices had been scaled successfully to large systems. In another case study, large-scale US government project followed the traditional software methodologies and, in this paper, we suggest how XP practices could be adopted in such organizations.

Additionally, a team from University of Sheffield assessed and compared XP with traditional method of software development. It was demonstrated that even if XP is not a plan-driven approach, it was able to obtain the final products that were of same quality and size as using traditional methodologies.

### 2.2.1 Order management System Architecture: It's a large and geographical distributed system

In the below architecture (Refer Fig.4) we have four micro-service architectures 1) Account Service 2) Product Catalogue 3) Cart Server 4) Order Server. Each of these carries a distinct database of its own and acts an independent system in own.

In the next level of architecture (Refer Fig.5), we can see that the requests or the orders from the client's end that can be from any of the browser from any of the devices does not hit the micro-services architectures directly but addresses the REDIS. This REDIS is cache that temporarily stores all the requests and orders are stored and accomplished. This practice gives you better performance as the response time delay will be minimized.

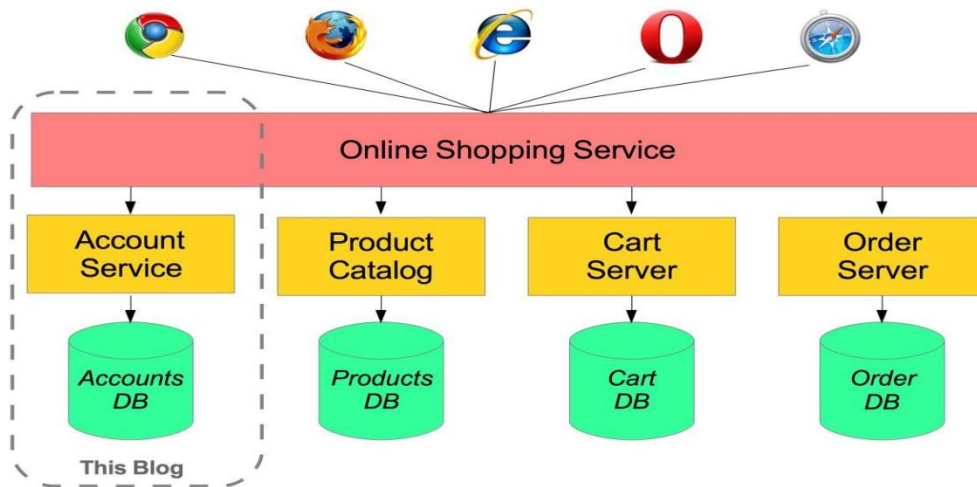


Figure 4: Order management System Architecture

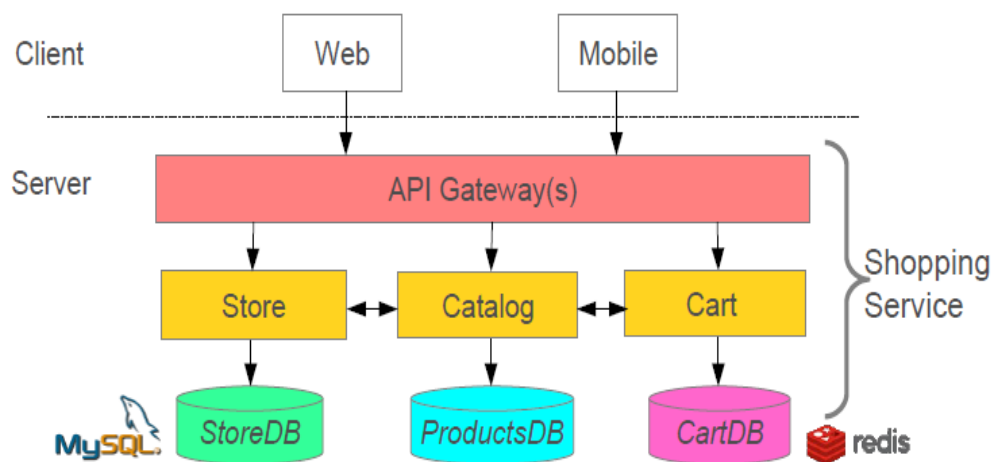


Figure 5: Mechanism of order management System

## 2.4 Extreme Programming Adaptation in a Large and Geographical Distributed Software System Settings:

This research conducted on XP proves that adaptation of XP in large-scale projects has effective team utilization. Both, qualitative and quantitative parameters were measured. The outcome of the overall project was satisfying as it was very quick in responding to changes in customer requirements. Both internal and external projects were compared. Both the teams have agreed to follow the recommended practices responding quickly to change, as there were no solid customers (Table 1).

Table 1: XP practices that were implemented in the projects.

Practices	Description
Daily standup meetings	Development team reflects for about 20 minutes in early morning before work commence on their activities.
Adaptive planning	Dependent teams work together to formulate and estimate user stories for iterations and releases
Code Control	For keeping project artifacts and source code organized and accessible.
Contentious Integration	At any moment, the Development team has to be ready with integrated working code that can be immediately tested.
Visual Indicators	Estimate what has been completed from user stories. They also include visual burn down charts.
XP Project management	During work teams communicate their user stories and report progress through a management tool Kanban/Jira/Confluence.
Code Gallery	Well-organized and innovative code is placed on the whiteboard to encourage creative thinking and increase refactoring skills.

Typically, the day of team members and developers start with a standup meeting followed with developers working towards implementation of system features either individually or in pairs depending on requirements of the user stories. Later, all source code is provided to code control tool and later to continuous integration tool to generate an executable code that is used for testing by the customer team. Lastly, refactoring is carried out by developers so that refactoring does not break the current system build.

Project management tool monitors the progress of team members through the metrics. A "code gallery" was derived from the XP practice that was incremental design in which the most coherent and innovational code was to be acquired and estimated and presented visually on the whiteboard. The working team agreed upon this condition. This approach allows every developer to contribute to the code gallery, hence encouraging the developers to improve their skills.

A challenging task was to develop as appropriate design that processes violations on such a heterogeneous, geographically distributed environment where the programmers are located at different



locations. Moreover, a team of developers along with the customers work in synchronization and this brought in another dimension of communication administration. The starting of project took some amount of time of the project to bring the team members in alignment of XP practices.

The team earlier worked only with a waterfall software development methodology and this was their first project with XP. Therefore, training on agile methodologies and migration to XP methodology was provided to this team

On project initiation, team was formed with five developers and four customer representatives. Both were trained and orientation was conducted Client/Server source code control tool "BitBucket", agile project management tool "JIRA/Confluence" and an uninterrupted integration server "Bamboo" was installed. The necessary infrastructure was provided for project code repository and connectivity to data centers. Project progress was tracked using Visual wall indicators like whiteboard, sticky notes and dashboards.

Development team members also played Subject Matter Expert based on their knowledge and experience. Technical Lead role was assigned to technically well-versed developer. The mentor role was played by project manager and he/she is also part of the development team. The mentor role was to mainly track user stories and maintenance of visual indicators. Also, to provide advanced technical supervision to the developer teams. From customer team, tester and mentor role were played to look into the quality assurance role. Project duration, iterations, code gallery etc were monitored and documented as in Table 1.

### 3. Results & Discussion

#### 3.1 Project Performance measures:

For 21 days sprint: Based on observations in software organization, we have done the below minimum case study analysis and full development plan. This case represents the 21 days development plan it includes the number of releases as per the team commits. The number of requirements, considering team size as 6 all having equal technical skills. TEAM SIZE: 6 (3 -Developers, 1-Tester, 1-Architect and 1-Project Manager)

The table 2 depicts the factors that represent the base of performance and evaluation of the system. Namely, the factors are: Numbers of Hours Spent in Training and Up gradations, Number of Iterations, Architectural Design Changes, Collaboration, Satisfaction, Interest and Velocity Increase these are directly proportional to the system performance.

Table 2: Full Development Plan

Numbers of Hours Spent in Training and Up gradations.	40 hours (8 Hours Each day for 5 Days).
Number of Iterations for 4 release	6 Iterations.
Application Architectural Design Change support	4-times.

Team Collaboration	High
Satisfaction(working environment/Culture)	High
Interest(buy-in)	High
Velocity Increase(Speed of deliverable)	4%
Code gallery	5 working Days(8 hrs/day)
Weekly work Hours	40 Hours (8 Hours/Week).

In the table 3 we can see that their releases and the code is exhibited four times in total. In table it's very clear that each time the code is exhibited the number of user stories completed are increased with an increased rate on each code exhibition.

Table 3: Release Plan

	Release 1	Release 2	Release 3	Total
Code Exhibit	1	1	2	4
User Stories Completed.	10	15	20	45
User stories Increase*	0%	1.5%	2%	3.5%
Story Points	140	220	290	650
Story Points Increase	0%	1.6%	2.1%	3.7%

In this section, the results are presented and summarized. The questionnaires in the survey were graded and are presented in Table4. User stories were collected and are represented using burn-down chart in Fig. 5. To steer of any ambiguity, questions were concise and very specific:

***“To what level or range does the following XP practices were realized during the project?”***

Respondents were given three options and they selected one among them, the three options were: implemented/realized, partly implemented/realized, not implemented/realized. If grade = 1, it implied no implemented was done, if grade = 2, it implied partly realized and for grade = 3, it implied that the practices were completely realized or implemented. It is shown in the results section.

Table4: Usage of Extreme Practices

XP practices	Average Grade
<ol style="list-style-type: none"> <li>1. Stories</li> <li>2. Informative workspace</li> <li>3. Whole team</li> <li>4. Energized work</li> <li>5. Slack</li> <li>6. Ten-minute build Incremental</li> <li>7. Design Continuous integration</li> <li>8. Sit together</li> <li>9. Real Customer Involvement</li> <li>10. Incremental Deployment</li> <li>11. Team Continuity</li> <li>12. Shared Code</li> </ol>	3
<ol style="list-style-type: none"> <li>1. Pair programming</li> <li>2. Weekly cycle</li> <li>3. Quarterly cycle</li> <li>4. Shrinking Teams</li> <li>5. Root-Cause Analysis</li> <li>6. Single Code Base</li> <li>7. Daily Deployment</li> </ol>	2
<ol style="list-style-type: none"> <li>1. Test-first programming Negotiated Scope Contract Pay-Per-Use</li> <li>2. Code and Tests</li> </ol>	1

### 3.2 User Stories burn -down Chart:

The burn-down chart below shows how the work is carried out each day. The chart in the picture shows the burn-down of twenty days starting from the “day 0” till “day 19”. The “day 0” starts with planning of 250 working hours. Each of the day is planned to work with certain hours and those hours are to be completed each day. These hours are deducted from the total number of hours remaining each day. For example, on “day 1” - 12 hours were planned but in actual 8 are completed. According to planned hours balanced hours must be 238 but actual balance is 242 hours. And the completed hours are 8. Each of the day further work is evaluated similarly.

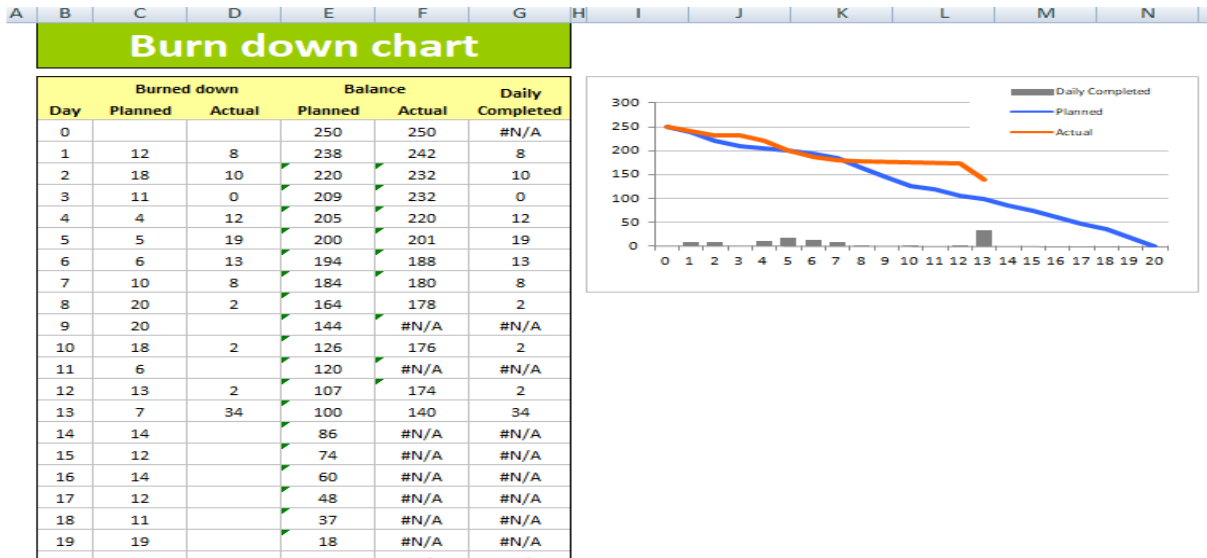


Figure 6: Burn-down chart

### 3.3 Comparison with legacy projects:

In below Table 5, a comparison of the online order management system was made with legacy projects like hospital management system, and lab management system. the result depicts that Project A which had a duration of two years took too much of time and Project B which had a duration of 6 months was up to expectation using the traditional system.

Table 5: Comparison of performance of different systems using Traditional System

Name	Team members	Durations	Comments	Result
Project A (Hospital Management System).	15	2 Years.	Large number of developers were working on requirement gathering.	Too Much Time
Project B (Lab Management System)	10	6 Months	Most of project time was in requirement gathering, and no single line of code was written.	Up to Expectations.

In Table 6, depicts a comparison of the online order management system was made with two different methods one is Crystal Clear Methodology and other is using extreme programming as well as cloud architecture, the order management system with XP and agile has good results and it requires a smaller number of team members and the duration is 12 months.

Table 6: Using Extreme Programming and cloud Architecture

Name	Team members	Durations	Comments	Result
Order Management System	12	18 Months	Completed on time, using Crystal Clear Methodology	Completed on time
Order Management System	10	12 Months	Completed before time using XP and agile.	Performance Increased Drastically

#### 4. Conclusion

XP programming which is suitable for small projects, with this study/model could be implemented to large and geographical distributed projects where development team are located across the globe. As a result of this process, the development of software becomes much faster, cost effective, loosely coupled, deployable across the globe. Extreme programming can be used for both developments: code and design centric and provides overall design of the software system. Overall, it is observed that Extreme Programming practices when adapted in the project increases the people-factor output and generates encouraging ideas about complex design issues.

#### REFERENCES

- [1] Aleti, A., Buhnova, B., Grunske, L., Koziolk, A., & Meedeniya, I., Software architecture optimization methods: A systematic literature review. *IEEE Transactions on Software Engineering*, 39(5), 2013, pp. 658-683.
- [2] Ramtin Jabbari, Nauman Bin Ali, Kai Petersen and Binish Tanveer, ‘What is Devops? A Systematic Mapping Study on Definitions and Practices’, the Scientific Workshop XP 2016.
- [3] Bingzhi Gao; Xiaojuan Ban; QiangLv; Xiaoli Li, A component-based method for software architecture refinement, *International conference on Intelligent Control and Information Processing*, pp. 574-578.
- [4] R.J.R Back, On Correct Refinement of Programs, *Journal of computer and system science* 23, pp. 49-68, 1981.
- [5] Elmuntasir Abdullah, El-Tigani, B. Abdelsatir, Extreme programming applied in a large-scale distributed system, *International Conference on Computing, Electrical and Electronics Engineering (ICCEEE)*, ISBN:978-1-4673-6231-3, pp.442-446, 2013.
- [6] Alexei Iliasov, Elena Troubitsyna, Linas Laibinis, Alexander Romanovsky, *Patterns for Refinement Automation*, ISBN: 978-3-642-17070-6, pp. 70-88, 2010.
- [7] Bin Xu, “Towards High Quality Software Development with Extreme Programming Methodology: Practices from Real Software Projects.” *International Conference on Management and Service Science*, pp.1-4, 2009.
- [8] DavidGarlan, “Software Architecture: a Roadmap”, School of Computer ScienceCarnegie Mellon University 5000 Forbes Avenue Pittsburgh, PA 15213(412) 268-5056

- [9] Kent Beck, “Extreme Programming Explained: Embrace Change”, Addison-Wesley, Reading, MA, 2000.
- [10] G. Vanderburg, “A Simple Model of Agile Software Processes –or-Extreme Programming Annealed”, ACM, 2005, pp. 539-545
- [11] Mrs. Nagalambika Swamy, Dr. L. Manjunath Rao, Mr. Praveen K S, Component Based Software Architecture Refinement and Refactoring Method into Extreme Programming, International Journal of Advanced Research in Computer and Communication Engineering(IJARCCE), ISSN: 278-1021, pp.398-401, 2016.
- [12] Nagalambika swamy, “Challenges In Software Architecture Refinement,” Research in Management and Information Technology, ISBN:978-1-5136-1658-2, pp.654-655, 2016.
- [13] Hong-Mei Chen, Rick Kazman, Serge Haziyeu "Agile Big Data Analytics Development: An Architecture-Centric Approach," Hawaii International Conference on System Sciences (HICSS), ISBN: 978-0-7695-5670-3, Koloa, HI, USA,2016.
- [14] E. Abdullah and E.-T. Abdelsatir, "Extreme programming applied in a large-scale distributed system," in Computing, Electrical and Electronics Engineering (ICCEEE), Khartoum, 2013.
- [15] Maleeha Arif Yasvi, “Review on Extreme Programming – XP”, International Conference on Robotics, Smart Technology and Electronics Engineering, Delhi, April 2019