“Challenges and Solutions in Distributed Software Development”

A case study from India on Hospital Information System built on OpenMRS framework.

Srinivas Krishna

A thesis submitted for the degree of Master in Informatics: Programming and System Architecture 60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO
Autumn 2023
“Challenges and Solutions in Distributed Software Development”

A case study from India on Hospital Information System built on OpenMRS framework.

Srinivas Krishna
Abstract

Nowadays, software development tends to be decentralized, putting more effort into developing areas that are more appealing to organizations. Distributed Software Development (DSD) is a type of software development in which team members are distributed across multiple locations and time zones. This method has the main benefit of making human resources more readily available in decentralized zones at a lower cost. However, the distance that separates the development teams presents some drawbacks in terms of technological advancement, culture, time zone, language, ownership, and many more. Coordinating and communicating effectively becomes increasingly challenging when software components are obtained from diverse sources, impacting project organization, control, and product quality. This necessitates the adoption of new processes and tools. As part of this thesis work, I would like to explore the current state of these challenges with a systematic review of the literature about the issues and solutions that have been proposed up until the present day in DSD and discuss the possible solution steps to overcome these challenges.
<table>
<thead>
<tr>
<th>Abbreviations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSD</td>
<td>Distributed software development</td>
</tr>
<tr>
<td>DHIS2</td>
<td>District health information software 2</td>
</tr>
<tr>
<td>HIS</td>
<td>Health information system</td>
</tr>
<tr>
<td>HISs</td>
<td>Hospital information system</td>
</tr>
<tr>
<td>HISP</td>
<td>Health information system program</td>
</tr>
<tr>
<td>UiO</td>
<td>University of Oslo</td>
</tr>
<tr>
<td>SDLC</td>
<td>Software development lifecycle</td>
</tr>
<tr>
<td>FVDIS</td>
<td>Free vertical domain information system</td>
</tr>
<tr>
<td>LMICs</td>
<td>Low and middle income countries</td>
</tr>
<tr>
<td>GSW</td>
<td>Global software work</td>
</tr>
<tr>
<td>GSA</td>
<td>Global software alliance</td>
</tr>
<tr>
<td>ICT</td>
<td>Information communication technologies</td>
</tr>
<tr>
<td>CAGR</td>
<td>Compound annual growth rate</td>
</tr>
<tr>
<td>OSS</td>
<td>Open source software</td>
</tr>
<tr>
<td>GSD</td>
<td>Global software development</td>
</tr>
<tr>
<td>WHO</td>
<td>World health organisation</td>
</tr>
<tr>
<td>UHC</td>
<td>Universal health coverage</td>
</tr>
<tr>
<td>PHC</td>
<td>Primary health centers</td>
</tr>
<tr>
<td>NHP</td>
<td>National health portal</td>
</tr>
<tr>
<td>HMIS</td>
<td>Health management information system</td>
</tr>
<tr>
<td>NDHM</td>
<td>National digital health mission</td>
</tr>
<tr>
<td>NHM</td>
<td>National health mission</td>
</tr>
<tr>
<td>EHRs</td>
<td>Electronic health records</td>
</tr>
<tr>
<td>NGO</td>
<td>Non governmental organisation</td>
</tr>
<tr>
<td>EMR</td>
<td>Electronic medical record</td>
</tr>
<tr>
<td>HIE</td>
<td>Health information exchange</td>
</tr>
<tr>
<td>OpenMRS</td>
<td>Open Medical record system</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Introduction**  
   1.2 Empirical Setting and Methods  
   1.3 Motivation  
   1.4 Expected contributions from the thesis.  
   1.5 Structure of the Thesis  

2. **Literature**  
   2.1 Introduction to DSD  
      2.1.1 Organizational Forms and DSD  
      2.1.2 Global Trends in GSW  
      2.1.3 DSD work involving India.  
      2.1.4 OSS work through DSD  
   2.2 Health Information System (HIS)  
   2.3 Hospital Information Systems (HiSs)  
   2.4 HISP and its Indian counterpart (HISP India)  
      2.4.1 HISP India  
      2.4.2 OpenMRS – Hospital  
   2.5 Dealing with open-source software (OSS)  
      2.5.1 OSS – Licenses  
      2.5.2 OpenMRS as OSS  
   2.6 Literature review analysis  

3. **Methodology**  
   3.1 Aim of the Thesis  
   3.2 Qualitative study methodology  
   3.3 Data collection methods  
   3.4 Timelines and Locations  
   3.5 Communication methods adopted.  

4. **Case study**  
   4.1 The object of study – The OpenMRS platform  
   4.2 The OpenMRS deployment in Himachal Pradesh  
      4.2.1 Key Objectives of the HISs in Shimla  
      4.2.2 Development Approach  
      4.2.3 Modules developed and implemented  
      4.2.4 Implementation location  
   4.3 Understanding of existing OpenMRS application architecture
Acknowledgments

First and foremost, I am extremely grateful to my supervisor, Prof. Sundeep Sahay for his invaluable advice, continuous support, and patience during my thesis. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. He expressed great working capacity with remarkable persistence and sufficient patience blended in, and the combination has benefited me tremendously. I would also like to thank Rashi Banta and Shravani Y for coordinating with HISP India team and helping me on my field visit to Himachal Pradesh. I am very grateful for their generous advice. And a big thanks to all UIO teaching staff and student administration team.

It would be unfair if I do not mention all the health workers, doctors, lab technicians, heads of the departments, and HISP India team that I interviewed during my field visit to gather data for my research work. I am very thankful to all of them for their valuable time and feedback they have provided. It is their kind help and support that have made my study and life in India a wonderful time.

I am very grateful to mention my father Dr. SM Krishna and mother Manjula GM for keeping their trust on me and to motivate me to pursue higher studies. I must extend my special thanks to my friends and well-wishers Priya, and Diler for being a moral support throughout my studies, my beloved brothers and sisters, Uncle Govindaraju, aunt Savithramma, my in-laws Siddegowda and Shivamma and all the family members for their heart felt blessings and prayers to smoothly complete my master’s course.

Finally, I would like to express my gratitude to my wife Priyanka Srinivas Krishna, and my children Taksh and Yuganth. Without their tremendous understanding and encouragement over the past few years, it would have been impossible for me to complete my study.
1. Introduction

Nowadays, software development tends to be decentralized, putting more effort into developing areas that are more appealing to organizations. Distributed Software Development (DSD) is a type of software development in which team members are distributed across multiple locations and time zones. This method has the main benefit of making human resources more readily available in decentralized zones at a lower cost. However, the distance that separates the development teams presents constraints in terms of technological advancement, culture, time zones, language, ownership, and many more. Distributed software development and distribution refer to the practice of developing and distributing software across multiple locations, teams, and stakeholders.

There is a tremendous demand for talent with the latest technology skills, business-side knowledge, and the ability to help companies implement digital transformation strategies. And as enterprises seek to acquire the skills and competencies for developing sophisticated software products and services, they also confront a severe shortage of tech talent. This DSD approach to software development has become increasingly popular in recent years as more companies have embraced remote work within global frameworks. In this approach, software development is done by geographically dispersed teams, using various tools and technologies to collaborate and communicate with each other. With distributed teams, organizations can arguably hire the best talent regardless of their location, allowing them to build diverse teams with varied skills and experiences.

As described in a research article by the technical research center of Finland, DSD facilitates software production independent of the physical location of individuals or organizations involved. Various business strategies, such as software subcontracting, partnership-based development, and global business ventures, and global software outsourcing exploit the benefits of this distributed approach. Despite these advantages, challenges persist, mirroring those faced by single-site projects. Quality, schedules, cultural difference, and cost-related issues are equally prevalent in distributed projects, with the added complexity of managing these challenges across different locations. Moreover, the geographical dispersion itself can contribute to time slippage problems, introducing delays and coordination difficulties. Therefore, while distributed development offers flexibility, it does not eliminate the traditional hurdles of software project management; instead, it amplifies their complexity.
Benefits of DSD

- **Access to global talent:** DSD allows organizations to tap into a global talent pool. By collaborating with developers and other team members from around the world, organizations can access a wider range of skills and expertise than they would be able to find in a single location.

- **Increased efficiency:** It allows teams to work concurrently across different time zones, leading to faster development times and quicker time-to-market by extending the working day (follow the sun).

- **Lower costs:** It can also help reduce costs for organizations. By leveraging remote workers and distributed teams, organizations can reduce their overhead expenses, such as office space and equipment, and avoid the high costs of hiring full-time employees in expensive locations utilizing the cheaper labour than what is available in western countries.

- **Increased flexibility:** It can also provide organizations with increased flexibility. With a distributed team, organizations can quickly scale their workforce up or down as needed, without having to worry about the logistics of hiring and firing employees.

- **Improved productivity:** When done right, distributed software development can improve productivity. By leveraging tools and technologies that facilitate remote collaboration, teams can work more efficiently and effectively than they would in a traditional office environment.

- **Greater resilience:** It can also provide greater resilience to organizations. By spreading out their workforce across multiple locations, organizations can reduce their risk of disruption from natural disasters, political unrest, or other events that could impact a single location.
• **Better customer service:** Finally, distributed software development can also improve customer service. By leveraging on a global talent pool, organizations can provide around-the-clock support to customers in different time zones, ensuring that they receive the help they need when they need it. Closer-to-the-market argument in DSD, when applied to customer service, emphasizes the advantages of having a diverse and geographically dispersed team to provide continuous and efficient support tailored to the needs of a global customer base.

DSD presents unique challenges\(^1\) when it comes to software design and upgrades. Here are some of the challenges that organizations may face.

• **Communication challenges:** Communication can be more difficult in distributed teams, especially if team members speak different languages or work in different time zones. Communication is key to successful software design and development, and this can be especially challenging in a distributed environment. Without the benefit of face-to-face communication, developers may struggle to collaborate effectively, leading to misunderstandings, delays, and mistakes.

• **Collaboration challenges:** Collaboration can also be challenging, as team members may not have the same level of access to tools and resources. With multiple teams working on different parts of the software, integration can be a major challenge. It can be difficult to ensure that different components of the software work together seamlessly, and that changes made by one team do not break the codebase for other teams.

• **Management challenges:** Managing a distributed team requires a different approach than managing an in-person team. Managers must be skilled at coordinating activities across other locations, time zones, and cultures.

• **Security challenges:** DSD can introduce security risks, especially if teams are not careful about transferring sensitive information.

• **Consistency:** With a distributed development team, it can be difficult to maintain consistency in software design and development practices. Developers may have different ideas about how to approach a particular problem, which can lead to inconsistencies in the codebase and make it difficult to maintain the software over time.

• **Upgrades:** Upgrades can be a major challenge with distributed software development. With multiple software versions in use across different locations, it can be difficult to ensure that everyone is using the latest version of the software and that upgrades are implemented smoothly.

Comprehending challenges in distributed projects is not only a struggle for global software development but also for research areas like OSS development and computer-supported cooperative work. Additionally, numerous papers in the literature explore distribution from specialized software engineering fields,
including requirements engineering and participatory design. Many economic and technological factors are driving companies to conduct projects on a global scale, resulting in highly distributed projects where experts from different companies, countries, and continents collaborate. Managing such distribution necessitates new techniques for project coordination, document management, and communication. Complexity arises from various project types, such as global, inter-organizational, or open-source software projects.

As per the literature analysis in one of the articles, researchers have discussed the distribution topics from several different perspectives, each of them having specific issues:

- Who or what is distributed?
- In what way are people and other entities distributed?
- What are the distribution's specific challenges?
- How can distribution challenges best be solved?

The below figure shows the interconnections among these perspectives, illustrating that distributed entities give rise to challenges, which, in turn, are addressed by specific solutions. Interestingly, these solutions can also introduce new challenges. For instance, Internet-based communication tools, as a solution, have the dual effect of creating and supporting distributed project settings.

![Figure 2. The cycle of distribution challenges](image)

DSD is a powerful approach to software development that can offer significant benefits for companies. However, it also comes with its own set of challenges, from communication and collaboration to management, design approach, and security. To overcome these challenges, organizations may need to invest in tools and processes that facilitate effective communication and collaboration across distributed teams. This may include using collaboration tools like video conferencing and chat, implementing code review processes to ensure consistency and quality, and using testing frameworks and automated testing tools to ensure that the software is tested thoroughly. Additionally, organizations may need to prioritize documentation and training to ensure that everyone on the team has a shared understanding of the software design and development practices being used.
While DSD offers many advantages, it comes with its own set of challenges. As discussed with many project stakeholders, successful implementation requires a combination of effective communication, the right tools, cultural awareness, and strong project management practices. As technology continues to evolve, addressing these challenges will become increasingly important for organizations looking to harness the benefits of a globally distributed workforce. In this thesis work, I am trying to explore the benefits and challenges of distributed software development and provide some key best practices for managing and improving this complex process.

As part of my further study, I would like to focus the areas like communication, collaboration tools, cultural differences, time zones, and project management part of the entire SDLC in a DSD framework. I have understood that the evolving landscape of technology will continue to shape how teams collaborate and overcome the challenges of distributed development. It's not just about having the latest tools, but also about fostering a collaborative mindset and adapting to the changing dynamics of a globalized workforce. As technology advances, we can expect improvements in collaboration tools, real-time communication, and perhaps even more sophisticated project management solutions. Artificial intelligence may play a role in automating routine tasks, allowing teams to focus on more complex aspects of software development. However, while the world is moving with these advancements in technology, we still see that some of the DSD implementations are still falling behind with legacy culture without adapting the new race. So, my main goal here is to observer such implementation and identify where the team's lack in adapting this technology during their sdlc process.

1.1 Research Objective and Research Questions

This thesis is based on my participation in a globally distributed development process of an Open-Source Health Information System. This type of software development scenario includes communication to share information, coordination of the group and its members, and planned activities and deliverables so that the overall effort contributes to achieving the project's overall goals. Distributed collaboration presents challenges to federated teams, such as delayed feedback, limited communication, reduced shared project awareness, difficulty in synchronized communication, inconsistent development and build environments, and lack of trust and trust between sites.

As stated in one of the research papers Distance Matters (Olson and Olson, 2000) challenges in DSD have been grouped into many categories.

- Collaboration readiness: People either must trust each other to do work of high quality and on time or they must set up contracts to ensure the work is done.
- Technology readiness: There is a wide range of kinds of technology that can assist in geographically distributed work, and if the team members...
have very limited experience, it can constrain the tools that might be used. Does the organization have a good system of technical support that can be called upon as needed?

- **Common ground (Shared understanding):** Effective collaboration requires that the participants have a common base of shared knowledge and vocabulary. This can be especially challenging for multidisciplinary collaborations. But participants in different locations also have very different contexts that can be a challenge for communication.
- **Nature of the work:** It is difficult to do tightly coupled or ambiguous work at a distance. Too much rich communication is required, and it will often take far longer to work out than if the tightly coupled work is done where people can work face-to-face.

With this understanding, as part of this thesis work, I would like to explore some key objectives of DSD lifecycle challenges and try to answer my 2 research questions.

1. What are the key challenges/impediments of distributed software development and distribution?
2. How do we address these challenges to improve the distributed SDLC and its distribution worldwide?

### 1.2 Empirical Setting and Methods

I started working on this thesis work taking one of the HISP India collaboration projects Hospital information System built under the open-source software OpenMRS platform.

**HISP India** is a non-profit organization dedicated to promoting the adoption and implementation of Health Information Systems (HIS) in India. HISP India focuses on leveraging technology to enhance healthcare delivery, improve patient outcomes, and contribute to the overall advancement of healthcare practices in the region.

**University of Oslo** is a globally engaged university located in Oslo, Norway, who helps students to develop knowledge in various streams that contributes to a sustainable world. I as a Master student am perusing the programming and systems architecture course in UiO approached my supervisor to do my thesis work on DSD.

**HISP** is a global action research project considered as one of the University of Oslo's greatest international successes. HISP has developed the DHIS2 software - an IT system for collecting, validating, analyzing, and presenting data for health information management activities. It has a global network around the world, HISP India is one of the branches of it. HISP India have many self-handled and paced projects where OpenMRS is one of its local initiatives.

**OpenMRS** is a collaborative open-source project to develop software to support the delivery of health care in developing countries. OpenMRS is founded on the
principles of openness and sharing of ideas, software and strategies for deployment and use. This is the area, that I have taken up for my case study.

I have taken a closer look at the software built by HISP India on this OpenMRS platform and have worked on one of the modules of the software to understand how the development process takes place. I conducted this study over a two-month period, where I connected with different team members who worked on the development activities and visited HISP India offices in Delhi and Shimla (Himachal Pradesh) along with some onsite visits to several hospitals identified in discussions with the state officials where the legacy Hospital Information System is deployed and is in use. To understand the aforesaid challenges and to discuss a few options to address those challenges.

My Field visits included:

**Delhi**
- HISP India office, Delhi

**Himachal Pradesh**
- HISP India office, Shimla.
- Deen Dayal Upadhyaya Zonal Hospital Shimla.
- Primary Health Center Shimla.

As a research student, my study is basically to understand the complexity of the design principles within a DSD setting. I created a checklist of questions and connected with the developers from HISP India and other software development organizations to gather information on best practices of DSD. I worked with the HISP India development team directly to gather the architecture details of the hospital information system, while getting the questionnaires filled by my identified respondents.

I have chosen the qualitative study methodology in my research, which required personal visits and observations and discussions with the developers, implementation team at the facility to understand the existing software development and deployment lifecycle, and the main challenges in the DSD process.

Qualitative research is a research method that collects non-numeric data. It goes beyond the information provided by quantitative research, as it is typically used to understand underlying reasons, opinions, and motivations.

The research questions presented in this thesis are investigated empirically within the framework of an ongoing project focused on the development of a Hospital Information System (HIS) application. This collaborative effort involved national distribution of workforce where developers were connected from different parts of India and with global presence, where I was working from Oslo, Norway.
The study aims to address and analyze the specified research inquiries in the real-world context of this collaborative venture, shedding light on the practical implications and outcomes within the domain of health information systems development.

Upon engaging with HISP and OpenMRS, it became evident to me that these projects, categorized as free vertical domain information systems (FVDIS), differ from my prior engagement and understanding of open-source projects. In this context, FVDIS refers to open-source organizational information systems designed for specific vertical domains, such as healthcare. This stands in contrast to horizontal infrastructure software like web servers and browsers.13

The unique characteristics of free vertical domain information systems, exemplified by projects like HISP and OpenMRS, prompted a reevaluation of my understanding of open source. The nuances in the development processes, community dynamics, and the targeted vertical domains distinguish these projects from the more generalized, horizontally focused open-source software that I had previously encountered and engaged with. This realization underscores the diverse nature of open-source contributions and the importance of recognizing the specialized requirements and contexts in which certain projects operate.

1.3 Motivation

The globalization of the workforce, as evidenced by the increasing trend in outsourcing for cheaper labor and specialized knowledge, has become an undeniable reality in recent years. DSD is an intriguing facet of this globalized approach, offering the ability to collaboratively work on a single software project from anywhere in the world, facilitated by internet connectivity or other network technologies that link team members. However, despite the apparent allure of this model, the practical implementation of DSD teams faces numerous obstacles. The promise of simultaneous global collaboration is met with challenges that range from communication barriers and differing time zones to cultural variations, posing intricate hurdles that demand thoughtful solutions for the effective execution of DSD projects.

My motivation to engage in this project is rooted in from my experience as a developer in DSD projects, primarily within commercial settings rather than public health. Notably, my lack of prior involvement in open-source projects within the health sector in Low- and Middle-Income Countries (LMICs) introduces distinctive challenges to this venture. It is this combination of unique project characteristics and the prospect of overcoming new challenges that fuels my enthusiasm and commitment to contribute to this research work.
1.4 Expected contributions from the thesis.

My thesis seeks to contribute to the following domains:

- To investigate the impact of geographic and cultural distance on communication and collaboration among DSD teams working on opensource development in a public health setting within a LMIC context.
- To explore the effectiveness of different tools and technologies for supporting communication and collaboration in DSD.
- To investigate the impact of distributed development on the software development lifecycle, including requirements gathering, design, development, testing, and deployment and to identify the challenges and opportunities associated with DSD in specific industry domains, such as healthcare.

1.5 Structure of the Thesis

Here is the structure of my thesis work going to be presented as part of this document.

1. Chapter 1: This chapter serves as comprehensive introduction to the methodological framework guiding the research questions, and motivation of this thesis work along with highlighting the structure of this thesis.
2. Chapter 2: This is the Literature chapter that discusses relevant literature.
3. Chapter 3: This is the Methodology chapter where I describe the methods that I have applied while working on this thesis to gather the data and analyze that data.
4. Chapter 4: This is the Case Study chapter where I discuss my case studies developed based on my empirical engagement.
5. Chapter 5: This is the case analysis chapter where I have compared the data, I have gathered to conclude my findings and discuss how I have responded to my research questions.
6. Chapter 6: This is the conclusion chapter where I have discussed my final opinions and future work.
2. Literature

In this chapter, an extensive review of the existing literature in the field of distributed software development is presented. This review serves to provide a comprehensive understanding of the current state of knowledge, challenges, and advancements within this dynamic domain. By synthesizing insights from a range of scholarly works, this chapter aims to lay the groundwork for a nuanced analysis of the complexities inherent in DSD. The literature discussed encompasses key themes such as communication and collaboration strategies, project management methodologies, technological tools and infrastructure, quality assurance practices, cultural and human factors, as well as emerging trends.

2.1 Introduction to DSD

DSD is the process of developing software by distributed teams across different geographic locations, often spanning multiple countries or continents. This approach has become increasingly prevalent due to globalization, advancements in communication technologies, and the availability of a diverse talent pool worldwide. Here are key aspects of global software development:

- Geographically distributed teams
- Time zone challenges
- Cultural diversity
- Communication and collaboration tools
- Project management strategies
- Quality assurance and testing
- Security and intellectual property concerns
- Legal and regulatory considerations
- Coordination and collaboration challenges
- Cost and resource optimization

DSD is often adopted to optimize costs and tap into a global talent pool. However, effective cost management and resource allocation are essential for successful outcomes. Despite its challenges, DSD offers the advantage of accessing diverse skill sets, cost savings, and the ability to work on a 24/7 development cycle. Successful DSD requires careful planning, effective communication, and the implementation of appropriate tools and methodologies.

DSD is a captivating phenomenon that has been a subject of research for the past three to four decades. The exploration of DSD in research has taken diverse approaches, covering various aspects of the subject. Researchers have delved into
the motives behind establishing DSD relationships, as highlighted by Herbsleb and Moitra in 2001. This line of inquiry seeks to understand the underlying reasons and incentives that drive organizations to engage in distributed software development.

Another significant area of research has focused on the challenges associated with DSD, as elucidated by Mockus and Herbsleb in 2001. This research stream aims to identify and analyze the obstacles and difficulties that emerge in the context of globally distributed software development efforts.

Furthermore, there has been research dedicated to capturing the experiences and learnings derived from DSD endeavors, as demonstrated by the work of Herbsleb, Paulish, et al. in 2005. This line of inquiry seeks to document insights gained, best practices discovered, and lessons learned from practical experiences in the realm of global software development.

Overall, the extensive body of research on DSD reflects a comprehensive exploration of the motivations, challenges, and lessons learned in the context of software development on a global scale. This research contributes valuable insights to the understanding and improvement of practices in the evolving landscape of global software development.

In the book Global IT Outsourcing: Software Development Across Borders, the phenomenon of global software work has been discussed in detail. The term «Global Software Work» (GSW) represents a form of DSD and is defined as the execution of software-related tasks carried out at geographically dispersed locations across national boundaries. This work is characterized by a coordinated approach involving real-time or asynchronous interactions. GSW encompasses activities conducted across global borders, facilitated through various arrangements such as outsourcing, alliances, or subsidiary relationships. Notably, GSW represents a relatively unexplored form of work, distinguished by organizational structures that differ from traditional global setups seen in large multinational corporations. GSW operates within a highly dynamic and diverse global marketplace, featuring organizations of varying sizes from both developed and developing countries. This arena is unique in that the success of firms is not necessarily hindered by their existing size. In essence, GSW provides opportunities for organizations, regardless of their scale, to actively participate in the globalized landscape and contribute to the field of software development.

2.1.1 Organizational Forms and DSD

This book also emphasizes the landscape of international business environments and organizational structures is undergoing significant transformation within what has been alternatively termed the 'new economy,' the 'digital economy,' the 'network
society,' or the 'information age.' In these emerging environments, notable changes are observed, particularly in the organizational forms adopted to facilitate global operations. A defining characteristic of these novel organizational forms is the way in which space and time have become pivotal considerations in reimagining the essence of an organization (Friedland and Boden, 1997).

An illustrative example of such a contemporary organizational structure is the 'Global Software Alliance' (GSA). This term is employed to encapsulate the unique nature of organizations created explicitly to facilitate Global Software Work (GSW). In these evolving contexts, the traditional notions of organizational boundaries and structures are redefined, reflecting a paradigm shift in how businesses operate on a global scale. The GSA, as a manifestation of this trend, signifies a departure from conventional organizational models and embodies the transformative impact of the digital age on global business environments.

The defining characteristics of GSAs include:

- The process involves physically separating different units within the network while electronically coordinating their activities across temporal, spatial, and cultural boundaries.

- GSA is no longer confined to large corporations with substantial financial capabilities. Small and innovative “born global” firms, propelled by technology, ambition, intellectual capital, and cost advantages, now actively engage in these alliances.

- Information and Communication Technologies (ICTs) play a central role in GSA operations, serving to coordinate activities among various work units and define the content of work.

2.1.2 Global Trends in GSW

Global Trends were also discussed in the same book where it points out that GSW’s diversity is expected to increase in the future predominantly. The authors write that the demand for and supply of GSW has increased substantially since the early 1990s with continued demand from well-established users in the USA, UK, Australia, and various Western European countries. According to the International Data Corporation, the USA is likely to continue to be the most significant user of GSW, with the predicted increase in spending from $5.5 billion in 2000 to a rather optimistic $17.6 billion by 2005. This prediction was less than reliable, we can see that the prediction was not at all overstated, as we see the current data as of 2022-2023 and the future prediction that is reported in precedenceresearch where the global IT services outsourcing market size was estimated at USD 525 billion in 2022 and it is
expected to hit around USD 1,149.24 billion by 2032, poised to grow at a compound annual growth rate (CAGR) of 8.2% during the forecast period 2023 to 2032. It also reports that the USA accounted for 42% of the revenue share out of this estimated market size.

My focus will be on understanding their specialization strengths and weaknesses, providing insight into the intricate complexity of this network. We’ll commence by examining the ‘big three’ nodes: India, Ireland, and Israel. Examining these global nodes and emerging centers underscores the diversity and complexity within the GSW network. Each location brings unique strengths and faces specific challenges, reflecting the dynamic nature of the global software development landscape.

2.1.3 DSD work involving India.

Though my research is on distributed software development and distribution across the globe, I am deep diving into Indian region on IT Outsourcing as my case study for this research has been done in India. Since the early 1990s, primarily driven by the goal of reducing personnel costs, numerous companies have ventured into multisite, multicountry software development strategies. Regions such as India and Eastern Europe have garnered significant consideration in this context. The approaches employed have spanned a spectrum, ranging from outsourcing segments of software development projects to third-party entities or subsidiaries, to the creation of genuinely virtual development teams.

In an article published in IEEE Software it is described how distributed development works with 4 models as shown in the below figure. Multisite and multicountry software
development can manifest in various forms through different models, each offering distinct advantages and presenting unique challenges. Among various factors, our experience indicates that the legal relationships between participating companies and the configuration of the teams are the most pivotal distinguishing features. It identifies four major cooperation models:

**Figure 5. The four cooperation models are according to company relationship and team setup.**

1. **Separate teams in basically independent companies**: This reflects the conventional contractor–subcontractor relationship. In this setup, various challenges arise across legal, knowledge transfer, development, project management, and quality management domains. When the distribution spans cultural boundaries, additional complexities may emerge, introducing challenges related to language, time differences, and infrastructure variations.

2. **Separate teams in legally related companies**: This involves essentially constituting a specialized contractor–subcontractor relationship between a mother and daughter company. This scenario shares similarities with Model 1; however, legal, knowledge transfer, and project and quality management issues are often easier to address because the mother company owns the subcontractor. This ownership structure helps facilitate coordination and communication between the teams.

3. **One team distributed across multiple sites of legally related companies**: emphasizing development and project- and quality-management issues. In this scenario, these issues take precedence. If the sites are distributed across different countries, additional challenges arise in the form of language barriers, time zone differences, and infrastructure disparities. Overcoming these challenges becomes crucial for the success of the collaboration.

4. The fourth model involves one team distributed across multiple sites of several essentially independent companies. This represents the most general mode of a globally operating company where teams are spread across various legally independent sites. This model encompasses all the implications of Model 3;
however, the legal implications can become more pronounced and present major challenges. Managing legal aspects along with the coordination of development, project management, and quality management becomes critical in this complex setup.

Table 1 presents how the four cooperation models align with the discussed criteria, providing a taxonomy for classifying collaborations. This taxonomy serves the purpose of identifying the relevance of lessons learned in different models. For example:

- In a collaboration following Model 3, insights and lessons from a Model 2 collaboration might be applicable, especially if the legal relationships between participating companies played a significant role.

- A collaboration structured according to Model 1 might benefit from lessons learned in a Model 2 collaboration, particularly if the rationale for separate teams was a key consideration.

The intent is to leverage lessons learned across different models based on the specific criteria and challenges associated with each model. This cross-model learning approach can enhance the effectiveness of collaborations and contribute to the improvement of global software development practices.
With the understanding from this case study about the collaboration between Indian firm and the German company, I take some lessons learned to my research work on how beneficial it would be of IT outsourcing across the globe especially to the main emerging entities of the IT outsourcing companies from India, China, Israel etc.

2.1.4 OSS work through DSD

From the Doctoral study from Nguyen Duc in 2015\textsuperscript{20} Distributed software development, global software development, and OSS are interconnected concepts within the realm of software engineering and project management. Here's how they relate:

- **Distributed Software Development:** This refers to the practice of developing software with teams located in different geographical locations. The development process is distributed across multiple locations, allowing organizations to tap into a global talent pool and benefit from diverse skill sets.
• **Global Software Development**: Global software development is a broader term encompassing the idea of developing software on a global scale. It includes both distributed development and any other approach that involves collaboration and coordination among teams located in different parts of the world.

• **Open-Source Software (OSS)**: Open-Source Software allows the source code of a software project to be openly accessible, modifiable, and distributable. It encourages a community-driven approach, where developers worldwide can contribute to the improvement and evolution of the software. It could be considered as the form of DSD as they frequently intersect in the development landscape. The collaborative and transparent nature of open-source projects aligns well with the principles of distributed development, making them complementary in many software development scenarios.

![Figure 6: Connection between DSD, GSD and OSS](image)

**Figure 6**: Connection between DSD, GSD and OSS

Relationships between DSD, GSD and OSS

• **Overlap**: Distributed software development and global software development often involve some degree of outsourcing. Teams in different locations or countries may be part of the same organization or could be external service providers.

• **Collaboration**: In all three concepts, collaboration is essential. Whether the teams are distributed internally within a company, globally across different offices, or externally through outsourcing, effective collaboration tools and communication strategies are crucial for success.
• **Resource Utilization:** Distributed and global software development, as well as outsourcing, are strategies employed to optimize resource utilization. They allow organizations to access specialized skills, reduce costs, and operate on a 24/7 development cycle by leveraging time zone differences.

• **Challenges:** Each approach comes with its own set of challenges. Managing communication, ensuring alignment with organizational goals, and dealing with cultural and time zone differences are common challenges across distributed software development, global software development, and outsourcing.

In this visual representation, the three models are interconnected to highlight their relationships. Distributed software development emphasizes decentralization within an organization, global software development extends this to a global scale, and open-source software development showcases a collaborative, community-driven approach where contributors can be distributed globally. Together, they illustrate the diverse ways in which software development can take place, embracing collaboration and contributions from various sources. In summary, these concepts are interconnected strategies employed by organizations to enhance flexibility, tap into global expertise, and optimize resources. The choice between them depends on factors such as project complexity, organizational goals, and the need for specific skills and expertise.

My research focuses on understanding DSD related challenges in the development of an open-source hospital information system project located in India.

### 2.2 Health Information System (HIS)

HIS is a comprehensive and integrated system designed to manage health-related information for individuals and populations. It encompasses a range of technologies, processes, and tools to collect, store, manage, and exchange health data. As described by WHO, the concept of Universal Health Coverage (UHC) is grounded in the idea that all individuals should have access to a comprehensive range of high-quality health services without facing financial difficulties. This access should be available when and where needed, covering the entire spectrum of essential health services. This spectrum encompasses health promotion, prevention, treatment, rehabilitation, and palliative care.

Achieving UHC requires countries to establish robust, efficient, and equitable health systems deeply rooted in the communities they serve. Primary Health Care (PHC) emerges as a pivotal and cost-effective strategy to realize the goals of UHC. Countries take unique pathways towards achieving UHC, determining the scope of coverage based on the specific needs of their populations and the available resources.
Investing in PHC becomes crucial in ensuring the identification, prioritization, and integrated addressing of diverse health needs. This investment also involves building a strong, well-equipped health and care workforce. Moreover, a successful UHC strategy involves the active engagement of all sectors of society. This comprehensive approach aims to address environmental and socio-economic factors that impact health and well-being. It encompasses preparedness for, response to, and recovery from emergencies, underscoring the importance of a holistic and resilient health system.

In summary, UHC is about creating a health system that leaves no one behind, with PHC playing a central role in achieving this vision. It involves tailoring health coverage to the unique context of each country, focusing on community engagement, workforce readiness, and a multi-sectoral approach to address the broader determinants of health.

An article published in May 2018 by Economist and Consultant from the Indian National Institute of Public Finance and Policy, New Delhi, describes in accordance with WHO, the Health Information System (HIS) serves as a critical foundation for decision-making in the field of healthcare, encompassing four key functions:

- **Data Generation**: HIS is responsible for the collection of data from various sources, including health and relevant non-health sectors. This involves gathering information on patient demographics, health conditions, treatment outcomes, and other relevant factors.

- **Compilation**: The collected data undergoes a compilation process within the HIS. This phase involves organizing and structuring the data in a systematic manner, making it ready for further analysis and interpretation.

- **Analysis and Synthesis**: One of the central functions of HIS is to analyze and synthesize the compiled data. This involves applying statistical methods, algorithms, and other analytical tools to derive meaningful insights. The goal is to transform raw data into actionable information that can inform decision-making processes.

- **Communication and Use**: The information generated through analysis is communicated to relevant stakeholders. This communication is a vital step to ensure that decision-makers, healthcare professionals, policymakers, and other stakeholders have access to the insights derived from the data. The aim is to facilitate informed decision-making and improve overall healthcare outcomes.
From the same article, it states that, UHC, now embraced by all UN member states as part of their sustainable development goals, has ushered in an era marked by enhanced governance and the progressive development of healthcare systems. The seamless operation, future growth, and sustainability of these healthcare systems depend significantly on the availability of complete and reliable information. The HIS plays a pivotal role in ensuring the recording, analysis, dissemination, and utilization of dependable and timely data by decision-makers across all tiers of the health system.

Information derived from HIS is utilized in a multitude of scenarios, ranging from the development of national strategies and plans to monitoring progress against established priorities. It is particularly critical in responding to public health emergencies. Additionally, reliable information is essential for fostering greater accountability for results, aligning with the commitment to UHC.

This discussion specifically delves into the current Health Information System of India, assessing its comprehensiveness and utility within the framework of international standards. The exploration encompasses various issues related to data collection, definition, analysis, and dissemination at the national, sub-national, and institutional levels.

India has been actively working on developing and HIS to improve healthcare delivery, management, and planning. Here are key aspects of health information systems in India:

- National Health Portal (NHP): The National Health Portal serves as a central repository for health-related information in India. It provides comprehensive information on diseases, health programs, medical facilities, and health tips.

- Health Management Information System (HMIS): HMIS is a flagship initiative under the National Health Mission (NHM) in India. It focuses on the digitization of health records, data collection, and reporting at various levels of the healthcare system. HMIS aims to improve the quality of health data for better decision-making.

- National Digital Health Mission (NDHM): Launched in 2020, the NDHM aims to create a national digital health ecosystem. It includes the development of Health IDs for individuals, a DigiDoctor platform for teleconsultations, and the Health Facility Registry for information on healthcare facilities.

- eHealth Initiatives: Various eHealth initiatives have been implemented to leverage technology for healthcare delivery. This includes the use of electronic health records (EHRs), telemedicine services, and mobile health (mHealth) applications.
• Telemedicine Services: Telemedicine services have gained prominence, especially during the COVID-19 pandemic. The government has encouraged the use of teleconsultations to enhance access to healthcare services, particularly in remote areas.

• State-Level Initiatives: Different states in India may have their own health information systems and initiatives. State-level HIS often integrate with national-level systems but may have specific features tailored to local needs.

• Public-Private Partnerships: Collaborations between the public and private sectors play a role in the development and implementation of health information systems. Private entities may contribute technology solutions and expertise to enhance the effectiveness of HIS.

• COVID-19 Response: The pandemic has accelerated the use of digital tools in healthcare. The Aarogya Setu app, for instance, was introduced for contact tracing and to provide information related to COVID-19.

Despite all these initiatives, the HIS in India faces challenges related to data quality, standardization, interoperability, and the integration of health information across different states and regions. Privacy and security concerns are also important considerations in the context of health data.19

In summary, the HIS cycle begins with the collection of data from diverse sources, proceeds with the compilation of this data into a usable format, moves on to the analysis and synthesis phase for meaningful insights, and concludes with the communication of these insights to decision-makers. Quality checks, relevance assessments, and ensuring the timeliness of data are integral components of this cycle, ensuring that the information derived from the HIS is accurate, meaningful, and ready for application in the decision-making process within the healthcare domain.

While working on this research work under the concept of HIS, I had to concentrate on the hospital information systems (HISs) which could be referred as a subset of a wider health information system. HISs particularly focuses on managing information within a hospital or healthcare facility, designed to address the unique needs and workflows of a hospital. It streamlines administrative, clinical, and financial processes. It facilitates efficient patient care, resource management, and communication among different departments within the hospital. Since, my case work is based on this hospital information system, I have discussed bit deeper in the next section. In the context of my case study related HISs, it has been implemented and used basically in secondary health care centers and tertiary healthcare centers.
2.3 Hospital Information Systems (HISs)

As seen in our previous section, HISs is an application developed to manage the medical, administrative, financial aspects of the hospital. It has evolved significantly to cater to the diverse needs of healthcare institutions and services. Various types of HISs, including commercial, non-commercial, closed source, and open source, are available in the market. However, the primary challenge in adopting HISs for health institutions lies in the complexity of customization to meet specific and local needs. In developing countries, the health sector grapples with management problems due to resource constraints. The adoption of HISs in these countries is hindered by budget limitations, leading to the utilization of either limited closed solutions or incomplete and complex systems. Open-source HISs emerge as a promising solution in this context, offering free and open access to source code. This characteristic simplifies adoption, especially in resource-constrained environments, and allows for better adaptation through modifications to the source code.

The objective of this work is to work with one of such open-source HISs which aligns with the specifications of healthcare institutions.

OpenMRS, being an Open-Source Software (OSS), serves this purpose as a versatile tool for collecting, validating, analyzing, and presenting aggregated statistical as well as event-based data. Its functionalities are crucial for the design and development of a HIS. The development of HIS on OpenMRS is an effort led by HISP India, Delhi and Shimla in particular, holds the responsibility for developing the core functionalities of HIS on OpenMRS.

HISP-India, a non-governmental organization (NGO), plays a pivotal role in the project under examination. With a track record spanning over 15 years, HISP-India specializes in public Health Information Systems (HISs) in India and other countries within the Asian region. The development team at HISP-India takes charge of customizing DHIS2, HIS on OpenMRS, and other OSS tools to meet the specific contextual needs of the project. Additionally, they provide essential capacity-building and support services to the project's customers, ensuring effective implementation and utilization of the HIS application.

In summary, the thesis investigates research questions within the context of a collaborative project between HISP India collaborating with global team for developing an HIS application. OpenMRS, as an OSS platform, serves as the technological backbone, with the HISP India team contributing to its core functionalities, and taking on the role of developer and provider of customized solutions and support services.

2.4 HISP and its Indian counterpart (HISP India)

Though it's a known entity and have internet knowledge base, I would like to record in my writings in this section about HISP in brief. HISP (Health Information System
Program) is a global action research project considered as one of the University of Oslo’s greatest international successes. It is a global action research network providing day-to-day direct support to DHIS2 users, including ministries of health and local implementers. DHIS2 (District Health Information Software) is one of the open-source HMIS (Health Management Information System) platforms developed by HISP and the world’s largest HMIS platform, deployed in more than 100 countries covering approximately 2.4 billion people.

2.4.1 HISP India

As part of my research work, I would be exploring more on HISP India, as that is where most of my empirical work has been conducted with respect to the open-source project, that is being developed and distributed under HISP India. HISP India is a key node in the global Health Information System Program (HISP) network.

HISP India’s work is in the domain of public health informatics, and specifically concerning the design, development, implementation, and support of DHIS2 based applications for governments and international organizations (WHO, PATH, CHAI and others). HISP India has a collaborative MoU with the University of Oslo (UiO) and jointly works on various DHIS2-related projects, including in the conduct of International DHIS2 related topics in India, HISP India trainers act as resource persons for other global academies in Africa and Asia. In addition to DHIS2, they also have experience in building non-DHIS2-based systems for Human Resources, Hospitals (based on OpenMRS), MHealth, ODK, and integrations across these systems. HISP India is one of the oldest and most proactive partners in the global HISP Network and has worked in Nepal, Bangladesh, Bhutan, Sri Lanka, Indonesia, and Tajikistan. Few of the major open-source-oriented products that HISP India is mainly working on are:

- **DHIS2 Aggregate**: It was developed by HISP India by customizing the DHIS2 features for aggregating the data such as state data warehouse and integrated analytical dashboard and reports.

- **DHIS2 Tracker**: DHIS2 customizations for disease registries, clinic-based systems, and the support of patient-level care.

- **DHIS2 Mobile**: DHIS2 Mobile serves as an extension of DHIS2-based solutions to cater to the information demands from outreach centers and peripheral clinics.

- **OpenMRS – Hospital**: To support integrated patient-based core within hospitals and clinical settings, HISP India has customized the OpenMRS platform to support different functions and has integrated it with the DHIS2 to enable aggregate facility reporting.
• iHRIS: customized iHRIS for human resources management and also integrated the data from iHRIS and HMIS to develop cross-cutting indicators to gain better insights into health workforce utilization and performance.

2.4.2 OpenMRS – Hospital

OpenMRS\textsuperscript{11} is both a software and a community. As a software, it serves as an electronic medical record system (EMR) originally designed for developing countries. OpenMRS, as an open-source health information system, finds applications in a variety of healthcare settings and scenarios. Its flexibility and adaptability make it suitable for a range of use cases. Here are some common applications of OpenMRS:

• Electronic Health Records (EHR): OpenMRS serves as a platform for creating electronic health records, allowing healthcare providers to digitize and manage patient health information efficiently.

• Clinical Data Management: OpenMRS helps in managing and organizing clinical data, including patient demographics, medical history, diagnoses, medications, and laboratory results.

• Health Facility Management: It is used for managing the operations of health facilities, including patient appointments, scheduling, and resource allocation.

• Disease Surveillance: OpenMRS can be employed for disease surveillance programs, aiding in monitoring, and tracking the spread of diseases within communities.

• Medical Research: OpenMRS supports medical research initiatives by providing a framework for collecting, storing, and analyzing data related to clinical trials and research studies.

• Public Health Initiatives: It is utilized in public health projects and initiatives, helping in the planning and execution of health interventions, vaccination programs, and other public health campaigns.

• Telemedicine: In telemedicine setups, OpenMRS facilitates remote access to patient records and enables healthcare professionals to provide virtual consultations.

• Humanitarian Healthcare: OpenMRS has been employed in humanitarian and disaster relief efforts to establish health information systems quickly and efficiently in emergency situations.
• Health Information Exchange (HIE): It supports interoperability between different healthcare systems, enabling the exchange of patient data across health organizations and systems.

• Primary Care and Community Health: OpenMRS is suitable for managing primary care services and community health programs, especially in resource-constrained environments.

• Education and Training: OpenMRS serves as an educational tool for healthcare professionals, allowing them to learn about health informatics and electronic health record systems.

• Custom Solutions: Organizations can customize OpenMRS to meet their specific requirements, making it adaptable to a wide range of healthcare scenarios.

OpenMRS's modular and open architecture allows for customization and integration with other health information technologies. Its global adoption in diverse healthcare settings underscores its versatility and effectiveness in addressing healthcare information management needs.

Through its open-source community, it has grown into a medical informatics platform used on every continent. OpenMRS is a framework built upon Java and other related frameworks. It is based on a modular architecture which consists of a core application and optional modules that provide additional functionality to the core workflows. The key architectural components of the OpenMRS core can be depicted as follows:
It all started by creating a basic data model, which was subsequently encapsulated within an API. This API-centric approach formed the foundation for a web-based application designed to interact seamlessly with the data model. The OpenMRS API operates as a "black box," concealing the intricacies of the data model beneath it. This abstraction ensures that applications and modules utilizing the API adhere to a consistent set of business rules for managing electronic medical record system data. With my interaction with HISP India team located at Delhi and Himachal Pradesh, I have gathered the knowledge of OpenMRS – Hospital has been developed and customized to vend the Hospital Information System to the state's public health sectors. It has been customized based on the Client and Server base model with AngularJS/ReactJS as a frontend client-side functionalities and Spring/Java-based server-side technologies as shown in the below figure.

At the heart of OpenMRS is a custom module framework that lets you extend and modify the default functionality of the OpenMRS core in accordance with your needs. Modules are also structured like the OpenMRS core and consist of the user interface, data access, and service layers. Some OpenMRS functionality is pulled out into modules instead of being written into the core application. This allows users to upgrade the content in those modules without having to wait for the next OpenMRS release.
2.5 Dealing with open-source software (OSS)

Since, as part of my thesis work, I am working mainly on the open-source software framework OpenMRS, it is important that I should gather some deep knowledge about OSS and its usage principles. Hence, I am going through some research areas on OSS and gathering some facts to use further in my case study.

In 1984, Richard Stallman founded the Free Software Foundation (http://www.fsf.org/fsf/fsf.html), a tax-exempt charity that raises funds for work on the GNU Project (http://www.gnu.org/gnu/thegnuproject.html). GNU is a recursive acronym for “GNU’s Not Unix” and a homophone for “new.” The GNU Project seeks to develop Unix-compatible software and return software to a state of freedom.

The article published by AltexSoft software R&D and Engineering found that 80 to 90 percent of today’s software is built out of open-source components. Having said that, there are 7 best practices that are discussed for managing open-source software development.
1. Prioritize a policy.
2. Update promptly.
3. Emphasize quality.
4. Use a binary repo manager.
5. Participate in the community.
6. Control with build tools.
7. Fork when possible.

Dealing with OSS involves various aspects, from using open-source software in your projects to contributing to open-source projects or even starting your own. A few key things to consider when dealing with open-source software are Understanding Open Source, Choosing Open-Source Software, Licensing, Contributing to Open Source, Community Engagement, Good Documentation, Security, Compliance, Support and Maintenance, Contribution Etiquette, Legal Considerations, Business Models, and need for staying Informed about the oss and its similar developments. One must also remember that open source is a collaborative ecosystem, and the principles of transparency, collaboration, and shared knowledge are at their core. By actively participating and adhering to these principles, we can benefit from and contribute to open-source software development and to the oss community responsibly.

While some question the value of open-source software development, its widespread popularity remains uncontested. This overview delves into the main principles of open-source licensing and development models. In contrast to proprietary vendors who adopt a closed-source model, develop software, release it to the public, and expect profit, the open-source movement, though still lucrative, operates on distinct practices. Open-source projects welcome anyone with coding capabilities, fostering robust software and diverse business models. However, managing numerous contributors necessitates comprehensive coordination, spanning from standardizing software to providing additional benefits. Even when packaged and sold by a third-party vendor, open-source software is cost-effective, liberating developers and hardware manufacturers from adhering to a closed-source software vendor’s specification. The ongoing efforts of the programming community contribute to the high reliability of open-source software. While free software offers unparalleled flexibility, stability, and freedom of choice, there is a tendency for various distributions to compete and mimic each other. This fragmentation is likely to pose the most significant challenge to the future of open-source software.

In the research paper by Ming-Wei Wu and Ying-Dar Lin, it is discussed that the open-source software development cycle, as illustrated in Figure 2 flow chart, enables broad participation, welcoming virtually anyone into the process. However, the involvement of numerous participants necessitates a substantial coordination
Developers employ various models to manage these extensive endeavors, ranging from the standardization of software.

![Diagram](image)

**Figure 9.** The general open-source system development cycle. Allowing multiple participants to contribute to the software development process requires a massive coordination effort.5

The landscape of open-source projects is vast, with a multitude in existence. Even though a significant number of these projects do not thrive, Fogel points out that many are, indeed, successful14. He emphasizes that the failures tend to receive less attention, as only the successful ones capture widespread notice. Given the sheer volume of free software projects overall, even if only a small percentage achieve success, the cumulative outcome is still a substantial number of visible projects14. This observation underscores the selective visibility of successful open-source projects and the challenges in highlighting the less successful endeavors within the expansive open-source ecosystem.

### 2.5.1 OSS – Licenses

The Open-Source Initiative has published a definition of what a license may and may not state to comply with the Open-Source Definition15. Few examples are:

- Free redistribution
- Access to source code
- No Discrimination
- Must allow modifications and derived works.
Similarly, the Free Software Foundation points to the Free Software Definition\textsuperscript{16} and its 4 kinds of freedom for the users of the software are:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works and adapt it to your needs.
- The freedom to redistribute copies so you can help your neighbor.
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

2.5.2 OpenMRS as OSS

OpenMRS\textsuperscript{17} (Open Medical Record System) is indeed an open-source software (OSS) project MPL 2.0 w/ HD. Being open source means that its source code is made available to the public, and users have the freedom to view, modify, and distribute the software.

Key characteristics of OpenMRS as an open-source project include:

- **Open-Source License**: OpenMRS is typically licensed under an open-source license, which governs how the software can be used, modified, and distributed. The choice of license ensures that the principles of open source, such as freedom and collaboration, are maintained.

- **Community Collaboration**: OpenMRS thrives on community collaboration. Developers, healthcare professionals, and volunteers from around the world contribute to the project. This collaborative model allows for the pooling of diverse skills and perspectives.

- **Transparency**: The source code of OpenMRS is transparent and accessible to anyone. This transparency fosters trust and allows users to inspect the code, ensuring security and enabling them to understand how the system functions.

- **Customization and Adaptation**: Users have the freedom to customize and adapt OpenMRS to meet their specific needs. This is crucial in healthcare settings where different regions and institutions may have unique requirements.

- **Global Impact**: As an open-source health information system, OpenMRS has been adopted and adapted for use in various healthcare settings around the world. Its global impact highlights the power of open source in addressing diverse healthcare challenges.
• **Support for Standards**: OpenMRS often adheres to health data standards, facilitating interoperability and the exchange of health information across different systems.

2.6 Literature review analysis

To conclude this literature chapter, I have gained a comprehensive understanding of DSD. Through a review of various research works and articles in this field, I have assimilated knowledge on what DSD entails, recognized the escalating growth and prevalence of distributed teams within the software industry, and discerned the distinctive challenges and opportunities inherent in distributed development. This literature review has provided valuable insights into the multifaceted landscape of DSD, equipping me with a solid foundation for further exploration and analysis in the subsequent sections of my research.

In reviewing the existing literature on DSD, a significant achievement has been the identification and categorization of challenges inherent in this approach. Various research works have diligently focused on key aspects such as communication, coordination, control, and complexity, shedding light on the common hurdles faced by distributed teams. The insights gained from these studies emphasize the critical need for effective strategies to address these challenges.

Most of the research conducted on the challenges of DSD consistently underscores key areas of difficulty, with a recurring emphasis on issues such as effective communication, cultural differences, coordination, and time zone disparities. Remarkably, despite the wealth of knowledge accumulated over time, these challenges persist as predominant contributors to failures in most DSD projects. The enduring nature of these obstacles underscores their significance and the ongoing need for innovative strategies and solutions to address them in the evolving landscape of distributed software development.

Furthermore, a noteworthy trend in the literature revolves around the intersection of agile methodologies with DSD. Several studies delve into the advantages and disadvantages of combining these two approaches, providing a nuanced understanding of their synergies and potential conflicts.

Building upon this wealth of theoretical knowledge garnered from the literature study, a valuable and complementary initiative involves active participation in a DSD project. By immersing myself in real-world DSD scenarios, I aim to gain hands-on experience and a firsthand appreciation for the challenges outlined in the theoretical domain. This practical engagement not only enhances the depth of my understanding but also serves as a foundational base for my forthcoming study on this intriguing and
evolving topic. The combination of theoretical insights and practical involvement positions me well to contribute meaningfully to the discourse on DSD challenges and strategies for effective management in distributed software development projects.

3. Methodology

In this chapter, I discuss the methodology adopted in my research. I have a qualitative approach to comprehensively investigate the challenges and solutions in DSD. It involves a case study, based on in depth interviews, document analysis, actual engagement in the development process and a survey to understand DSD related practices in different organization. Participating candidates were from global enterprise organizations with experience in distributed collaboration.

3.1 Aim of the Thesis

As a master’s student of IFI (department of informatics) from the University of Oslo, Norway, I have taken this thesis topic of “Challenges and Solutions in Distributed Software Development.” The rationale of this thesis comes from the fact that while the technology has reached its peak in the development and management of software systems, the management of distributed software development is constrained by multiple challenges and difficulties. I conducted my primary empirical work in the states of Himachal Pradesh and Delhi.

I conducted this study over a period of 8 months including my literature study, and was involved in development activities, field visits, data analysis, and documenting the thesis. I spent 4 months from August 2022 to November 2022 on OpenMRS – Hospital product upgrade project with the HISP India team and a field visit of over a week during January 2023 in which I could visit the developers, testers, and Implementation teams in Delhi and Shimla. I also visited a few hospitals identified in discussions in Shimla, to get hands-on experience on the OpenMRS – Hospital systems installation, configuration, and operations procedure along with observing and interviewing the end users.

During my in-person hospital visits, I spent 2-3 hours in each hospital and sought to meet the Doctors, the microbiology lab technicians, and others who were related to the maintaining and implementation of OpenMRS - Hospital systems. I had a predefined checklist of questions to ask, with minimum disruption to their work. All interview times were previously agreed upon, and all responses were fully anonymized.
3.2 Qualitative study methodology

Qualitative research\textsuperscript{21,22} is a research method that collects non-numeric data. It goes beyond the information provided by quantitative research, as it is typically used to understand underlying reasons, opinions, and motivations. I have chosen the qualitative approach to better follow the exploring development process to understand the situation and investigate the problem in question with an open mindset. The qualitative method can evoke responses that are unanticipated by the investigator and therefore well-suited to identify new problems. It also allows the adaptation of the research design according to the learnings with its flexible nature. Qualitative research is usually descriptive in nature and is often used in social research as it provides insights into people's behavior, relevant in health services research, where the phenomenon is complex and requires building context-based understanding. It helps examine the how and why of the decision-making processes, rather than when what, and where of the underlying problem analysis. Qualitative research approaches typically involve:

- Assuming dynamic and multiple realities.
- Performing In-depth interviews with semi-structured questionnaires.
- Conducting focus group discussions and observations.
- Concentrating mainly on the process over visible outcomes.
- Building case studies of multiple sites to build intra and inter-site analysis.

3.3 Data collection methods

I used multiple methods for data collection, including:

- Document analysis: By studying the OpenMRS – Hospital upgrade projects documentation available from OpenMRS portal and from HISP India team as listed below.
  - All modules data fields excel sheet.
  - All modules API documentation.
  - HP User manual.
  - Hospital Information System in Shimla Himachal Pradesh.\textsuperscript{39}
  - https://wiki.openmrs.org/
- One-on-one Interviews: Conducting around 24 interviews spanning a total of 66 hours. with HISP India team, medical college Doctors, Lab attendants, hospital administrators, and health workers.
- Observations: by visiting the primary and tertiary health centers and observing how they use the system and use the system for tracking and reporting purpose.
- Engaging in software development work by working with HISP India team on the OpenMRS – Hospital application where I worked on reports module of the application.
Online Survey: Conducting an online survey with many DSD exposed software professionals from global enterprises.

A Summary of the interviews conducted is summarized in the table below.

<table>
<thead>
<tr>
<th>Sl No</th>
<th>Designation</th>
<th>Place</th>
<th>No of Interviews</th>
<th>Issues Discussed</th>
<th>Total Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Backend Developer</td>
<td>Bangalore, HISP India</td>
<td>4</td>
<td>Legacy application architectural design issues and Attrition rate of team members which vaporizes the knowledge.</td>
<td>10 hrs.</td>
</tr>
<tr>
<td>2</td>
<td>OpenMRS Coordinator</td>
<td>Mumbai, HISP India</td>
<td>5</td>
<td>Project management and co-ordination issues.</td>
<td>10 hrs.</td>
</tr>
<tr>
<td>3</td>
<td>Frontend Developer</td>
<td>Delhi, HISP India</td>
<td>2</td>
<td>Co-ordination and source code version control issues.</td>
<td>10 hrs.</td>
</tr>
<tr>
<td>4</td>
<td>OpenMRS/DHIS2 Architect/Senior Developer</td>
<td>Delhi, HISP India</td>
<td>2</td>
<td>Legacy application architectural design issues and Attrition rate of team members which vaporizes the knowledge.</td>
<td>8 hrs.</td>
</tr>
<tr>
<td>5</td>
<td>OpenMRS/ Senior Developer</td>
<td>Delhi, HISP India</td>
<td>3</td>
<td>Application Integration and deployment issues.</td>
<td>6 hrs.</td>
</tr>
<tr>
<td>6</td>
<td>OpenMRS – Hospital Implementation Coordinator</td>
<td>Shimla, HISP India</td>
<td>2</td>
<td>Procurement and implementation issues</td>
<td>4 hrs.</td>
</tr>
<tr>
<td>7</td>
<td>OpenMRS – Hospital Implementation Team</td>
<td>Shimla, HISP India</td>
<td>2</td>
<td>Not possible to working remotely</td>
<td>10 hrs.</td>
</tr>
<tr>
<td>8</td>
<td>Patient Registration receptionist - Deen Dayal Upadhayya Zonal Hospital</td>
<td>Shimla</td>
<td>1</td>
<td>Application slowness and network issues due to on-premises implementation.</td>
<td>2 hrs.</td>
</tr>
<tr>
<td>9</td>
<td>General Physician - Deen Dayal Upadhayya Zonal Hospital</td>
<td>Shimla</td>
<td>1</td>
<td>Turnaround time is being big from operations team during the system unavailability.</td>
<td>2 hrs.</td>
</tr>
<tr>
<td>10</td>
<td>Health Nurse – Primary health care center</td>
<td>Shimla</td>
<td>1</td>
<td>Application slowness and network issues due to on-premises implementation.</td>
<td>2 hrs.</td>
</tr>
<tr>
<td>11</td>
<td>Senior General Physician-Primary health care center</td>
<td>Shimla</td>
<td>1</td>
<td>Need of faster application and application to be accessible by wider population.</td>
<td>2 hrs.</td>
</tr>
</tbody>
</table>

3.4 Timelines and Locations

I spent time in understanding the new Hospital product is being redesigned and developed for Himachal Pradesh state government hospitals and institutes. I started working on one of the reporting features to the upgrade project. I had a field visit of over a week during January 2023 in which I could visit the developers, testers, and Implementation teams in Bangalore, Delhi, and Shimla. I have also visited a few hospitals identified in discussions in Shimla, to get hands-on experience on the OpenMRS – Hospital systems installation, configuration, and operations procedure along with observing and interviewing the end users.

Oslo, Norway

In August 2022, I was granted the opportunity to work with the HISP India team. I started this journey with online collaboration, while I was doing my last semesters of
master courses. Over the course of four months, I established a solid foundation for understanding OpenMRS, HISP India and its operations. During this period, my focus shifted towards comprehending and contributing to the development and customization of the OpenMRS-Hospital application. I actively participated in software design, development, testing and presentations and engaged in formal and informal meetings with various team members.

**Bangalore, India**

On December 25th, 2022, I embarked on my field visit, and upon reaching Bangalore, I engaged in meetings with one of the lead developers involved in the backend system integration with the OpenMRS-Hospital application. These meetings were instrumental in gaining insights into his perspective on the integration process, particularly viewed through the lens of distributed software development. Our discussions delved into the intricacies of the OpenMRS-Hospital application, exploring its integration aspects within the context of a distributed software development framework.

**Delhi, India**

On January 9th, 2023, I journeyed to Delhi with the purpose of meeting the key development team at HISP India. Over the course of three days, I engaged in discussions with various team members, including a senior DHIS2/OpenMRS developer, a Frontend Developer, a Tester, and a DHIS2 mobile App developer. During these meetings, I had the opportunity to present my research points and gather their valuable opinions regarding the distributed software development aspects related to my research. These discussions provided diverse insights and perspectives, enriching the understanding of the distributed development landscape within the context of DHIS2 and OpenMRS.

**Shimla, India**

On January 12th, 2023, I traveled to Shimla in the state of Himachal Pradesh with the objective of meeting the OpenMRS – Hospital application Implementation team and the actual users of the application. During my three-day stay, I had the opportunity to engage with individuals from both the HISP India team and the hospital staff. The primary focus of these interactions was to gain a comprehensive understanding of the challenges faced by the implementation team and the end-users of the OpenMRS – Hospital application. These firsthand insights proved invaluable in shaping a more nuanced perspective on the real-world implications and hurdles within the application's operational environment.

Himachal Pradesh, situated in the northern part of India, is home to an approximate population of nearly 8 million. The state government of Himachal Pradesh has been
at the forefront of progressiveness, ensuring effective health services encompassing prevention and treatment. The Health and Family Welfare department in the state is dedicated to delivering a spectrum of services, including curative, preventive, promotive, and rehabilitative, through an extensive network of healthcare facilities.

The health infrastructure comprises 75 hospitals, 87 Community Health Centers, 533 Primary Health Centers, 13 ESI (Employee State Insurance) Dispensaries, and 2,078 Sub-centers. In its commitment to enhancing healthcare services, the government is actively reinforcing the existing infrastructure by incorporating modern equipment and specialized services. Efforts are also directed towards augmenting the medical and para-medical staff in various medical institutions.

For data collection, I conducted personal visits to three healthcare facilities within the region, gathering valuable insights into the healthcare landscape, challenges faced, and the ongoing efforts to improve and expand healthcare services in Himachal Pradesh.

- Deen Dayal Upadhyay Public Hospital, Shimla
  a. Reception for registration
  b. General practitioner
  c. Orthopedic
  d. Laboratory
  e. Pharmacy
- Primary health center
- Tibetan health center

Figure 10: Research Timeline
3.5 Communication methods adopted.

**Digital communication:** The advent of online communication has sparked a revolution in recent years, proving particularly beneficial in scenarios where customer and development teams find themselves geographically separated. This project exemplified the advantages and challenges of coordinating a team with members in Norway and various cities in India. A range of online communication tools, including Zoom meetings, email exchanges, WhatsApp group collaboration, and live chats, facilitated seamless communication among distributed team members. User manuals were shared online, enhancing participants’ preparation before group sessions.

**Face-to-face meetings:** I have conducted around 24 face to face meetings spanning around 66 hours of discussion. This played a crucial role in fostering mutual understanding among project stakeholders and addressing issues in real-time. These meetings offer a shared platform for distributed teams, allowing participants not only to delve into technical aspects but also to develop social understandings with one another.

This in-person interaction facilitated the identification of weaknesses in previous approaches and offered insights for potential improvements in the future. Despite its effectiveness, such face-to-face meetings incur high costs and cannot be arranged regularly. Consequently, this meeting was designed as a one-off event, acknowledging the practical constraints associated with frequent in-person gatherings.

I conducted one-on-one interviews with a diverse range of professionals, including developers, architects, testers, health nurses, general practitioners, and clinic attendants. This approach allowed me to gather firsthand insights into the DSD process and its implementation from individuals representing various roles within the project. An interview, as defined by Oates (2005), is a conversation where one party seeks to acquire information from the other. Interviews can be classified into three types: structured, unstructured, and semi-structured interviews (Cornford and Smithson, 2005; Oates, 2005). A structured interview adheres to a predetermined sequence of pre-planned questions, while an unstructured interview does not rely on pre-planned questions and delves into in-depth topics that emerge during the conversation with the interviewee. Semi-structured interviews, the focus of this study, follow a predefined interview guide outlining key questions but also allow the interviewer to explore in-depth emergent topics, resembling the flexibility of unstructured interviews. For this study, semi-structured interviews served as the primary data collection technique.
Online surveys: I conducted an online survey approaching over 100s of my software development life cycle exposed friends, colleagues, and professional network. However, I could manage to get 24 responses to my survey requests, but with very good quality responses. I believe having quality data over the quantity as its more quality results we see with better quality data inputs. This offered a versatile and efficient means of gathering data from diverse audiences, making them a valuable tool for research, market analysis, and decision-making in various fields.

Responders for my survey, was from 17 different software companies and around 20 different designated roles as listed below, and all the responses will be attached as part of the Annexure A.

Responders organisation

- Altera
- Amazon
- DNB ASA
- GEC
- HPE
- Infosys Limited
- Kanari AS
- Kuhne Nagel GmbH
- NGI
- Salesforce
- Tata Consultancy Services Ltd
- UHG
- Vipps
- Freelancer

Responders designate roles

- Cloud Architect
- Faculty
- Software Engineering - LMTS
- Devops Engineer
- Head of technical platforms
- Design Engineer
- Senior Tech Arch
- Principal Architect
- Sr Director software engineering
- Architect
- Freelancer
- Principal Consultant
- Software Developer
• Solution Architect
• EDI specialist
• CloudEngineer
• Software Engineer
• Devops Engineer Cloud
• Product Manager
• Team Lead - Service Delivery Manager

4. Case study

In this chapter, I present my case study based on the empirical work carried. This chapter includes the following sub-sections, where I highlight the object of the study the OpenMRS platform, its implementation in Himachal Pradesh, Understanding the legacy application, Need for the upgrade and my hands on involvement in this upgradation project.

4.1 The object of study – The OpenMRS platform

The OpenMRS – Hospital application was initially deployed in a standalone architecture using OpenMRS version 1.x in 2010. As the system evolved over time, the need arose for a substantial upgrade. This upgrade aimed to transition the existing standalone system into a web-based architecture, enabling the utilization of API functionalities, seamless integration with multiple other systems, modernization of existing features, and the establishment of compatibilities with National Health Authority (NHA) standards. This transformation was driven by the imperative to enhance the overall functionality, interoperability, and compliance of the OpenMRS – Hospital application with contemporary healthcare standards and practices. As part of this upgrade, I got a chance to contribute with my thesis work by working on one of the functionalities in this new upgraded version of the application.

With my thorough study of the architectural design and the functionalities, below is the OpenMRS System data flow, which consists of 3 layers such as web/frontend layer, App layer and the data layer as shown in the below figure.

![Figure 11: OpenMRS System Data Flow](image-url)
As shown in the above figure of the architectural design and functionalities, the OpenMRS System data flow can be outlined as follows:

**Web/Frontend Layer:** Designed and developed using ReactJS. This layer serves as the user interface, where interactions and engagements with the OpenMRS system take place. The use of ReactJS ensures a responsive and dynamic frontend experience.

**App Layer:** Developed in Java. This layer encapsulates the application logic, providing the core functionalities of the OpenMRS system. Exposed Java APIs facilitate communication and interaction with the frontend layer and other components.

**Exposed Java APIs for OpenMRS:** The Java APIs exposed by the App Layer enable communication between the frontend and backend, allowing for the seamless execution of operations, data retrieval, and other interactions within the OpenMRS system.

**Data Layer:** Utilizes MySQL. The data layer is responsible for storing and managing the system's data. MySQL serves as the relational database management system, ensuring efficient and organized storage of information.

OpenMRS System's data flow is orchestrated through a well-defined architecture comprising a ReactJS-powered Web/Frontend Layer for user interactions, a Java-based App Layer with exposed APIs for core functionalities, and a MySQL-backed Data Layer for efficient data storage and retrieval. This architecture ensures a cohesive and effective flow of data and operations within the OpenMRS System.

### 4.2 The OpenMRS deployment in Himachal Pradesh

In India, District Hospitals cater to a large population from as low as 32000 per district to as high as 30 lakhs per district\(^3\)\(^9\), providing variety of services, and absorb a good amount of financial and human resources. Despite these, their performance measurement & monitoring remained a challenge in public health systems. Traditionally hospitals were working on paper-based records and were reporting aggregate numbers on limited parameters in the Health Management Information System (HMIS). The information reported from district hospitals was limited to the
service delivery events required by National Health Programs. Information on curative care and administration required for facility management was remained in the hospital registers due to a lack of a proper recording & reporting system. Electronic Medical Record Hospital Information System has the potential to strengthen the clinical, management, and administrative systems in the hospital. However, these systems are difficult to implement and have their challenges and limitations as faced by ‘resource-rich’ Western hospitals. In India, District Hospitals function with a certain level of complexity and their information requirement changes very frequently. For an information system to be successful in that setting it is desired that it should have greater flexibility to adapt to the changing needs of the health system.

4.2.1 Key Objectives of the HISs in Shimla

Himachal Pradesh State implemented an OpenMRS-based Hospital Information System in District Hospitals with the help of NHSRC & HISP India in the year 2010. The objective of this initiative is to enhance healthcare services by documenting longitudinal patient records and providing aggregate data for hospital administrators and national program reporting. Key objectives of this deployment included:

Clinical Objectives
- **Integrated patient records**: Provide integrated and comprehensive patient records that span over time.
- **Improved service delivery**: Facilitate better service delivery through integrated operational workflows that share a common data structure.

Administrative Objectives
- **Access to operational information**: Provide access and availability of operational information to monitor the performance of various units within the hospital.
- **Decision-making support**: Improve decision-making processes by making operational information readily available.

Managerial Objectives
- **Hospital performance reports**: Provide managerial reports on hospital performance, including quality indicators and revenue generation.
- **District health system reporting**: Generate reports required for district health system reporting.
4.2.2 Development Approach

**Participatory method:** Adopted a participatory method for system development, involving ongoing collaboration with stakeholders.

**Requirements gathering:** The HIS development team documented requirements through a participatory process involving stakeholders.

**Mock-Up screens:** Converted requirements into mock-up screens to visualize the system’s design.

**User feedback:** Reviewed mock-ups based on user feedback, ensuring that the system aligns with the needs of the healthcare professionals.

4.2.3 Modules developed and implemented.
HIS System requirements collected from different stakeholders by HISP India team is to develop and customize the HIS – OpenMRS system with the following working modules.

- Registration module
- Billing module
- Laboratory module
- OPD
- Inventory module
- Pharmacy module
- IPD
- Blood Bank module
- Radiology
- Reporting module

4.2.4 Implementation location
With the overall goal of creating a system that considers patient flow, workflow, and information flow within the hospital, leading to more effective and efficient healthcare services. The project was initially developed and implemented in Deen Dayal Upadhyay Zonal Hospital, Shimla, and later replicated the setup in other primary and tertiary healthcare centers.
The development and implementation of the integrated hospital information system, which commenced at DDU Hospital in Shimla, has successfully expanded to encompass five additional hospitals: RH Solan, DH Kulu, ZH Mandi, DH Hamirpur, and DH Dharamshala, as well as RHP Medical College and Hospital, Tanda Kangra. Soon, the plan is to extend this system to cover more than 12 such healthcare facilities. This system has not only streamlined patient record-keeping to ensure continuity of care but also facilitated the generation of aggregate reports through patient-based encounters, thereby enhancing data integrity and quality. Furthermore, it has rationalized complex hospital processes, leading to improvements in data collection and reporting. The implementation process has also focused on building local capacity to manage the Hospital Information System effectively.

4.3 Understanding of existing OpenMRS application architecture.
In Himachal Pradesh, HIS has been implemented with few public hospitals discussed in the previous section, where the system has been locally installed and maintained locally within the hospital premises as captured in the below architectural diagram. Existing architecture consists of 3 different methods of deployment.

- Hospital portal
- Online registration portal
- MPI reports portal

![Figure 12: Existing HIS Implementation](image-url)
4.3.1 Hospital portal (locally deployed and maintained)

HIS software is locally deployed in a hospital premises where the software application and associated data are installed and run on servers or computers within the physical boundaries of the hospital. Existing portal looks as in the below screenshot.

- Patient records, medical histories, and other healthcare-related data are stored on servers within the hospital. This allows for direct control over data management and security.
- Healthcare professionals within the hospital access the HIS software using computers or devices connected to the hospital's internal network. Access is typically restricted to authorized personnel.
- Locally deployed HIS software is integrated with other hospital systems, such as laboratory information systems, radiology information systems, and pharmacy systems, to provide a comprehensive view of patient information.
- Hospitals often have specific requirements and workflows. Locally deployed HIS software allows for a higher degree of customization to adapt to the unique needs of the hospital.
- In the event of network disruptions, locally deployed HIS systems may still allow certain functionalities to be accessed offline. This is crucial for ensuring uninterrupted healthcare services.
- Software updates, patches, and maintenance tasks are managed locally by the HISP India team. This gives the hospital more control over the timing and implementation of updates.
• Hospital staff is typically trained on-site to use the HIS software. The hospital's IT support team helps and resolves issues directly.

• While locally deployed HIS systems offer control and customization, they also require a significant investment in IT infrastructure and ongoing maintenance. This in turn becomes an IT bottleneck and operations challenge.

4.3.2 Online registration portal (centrally managed)

A centrally managed HIS registration portal is a component of the broader HIS used in HIS – OpenMRS Portal. The registration portal is a system that manages the registration and admission of patients into the hospital. It is designed to streamline and automate the registration process, ensuring accuracy of patient information and providing a centralized database for the public healthcare facilities. As part of this portal, patients from different parts of the state can register online and book an appointment with the appropriate hospitals and the doctors available near to them. Later this registration details have been posted to the individual hospital’s portal for further processing. Existing Patient Registration Portal looks like as shown in the below Figure.

- It captures and stores patient demographic information, including name, age, gender, address, contact details, and emergency contact information.
- It assigns a unique identifier to each patient to ensure accurate and unambiguous identification throughout the hospital system.
- Integrate with the appointment scheduling system to facilitate the registration of patients for scheduled visits, tests, or procedures.
- Manage the admission and discharge processes efficiently, including assigning beds, recording admission details, and generating discharge summaries.
- Facilitate the completion and storage of consent forms required for medical treatments and procedures.
- Capture and update billing information, including payment methods, cost estimates, and co-ordination with other systems.
- Seamlessly integrate with the broader HIS, including the electronic health records system, to ensure a unified patient record.
- Generate and print patient identification wristbands with relevant information for easy identification during the hospital stay.
- Implement robust security measures to safeguard patient information. Ensure secure user authentication for healthcare professionals accessing the registration portal.
- Maintain an audit trail of changes made to patient records to ensure accountability and compliance with data protection regulations.
- Provide reporting tools and analytics to monitor registration trends, patient demographics, and other relevant metrics for administrative purposes.
This Patient registration portal is available on the locally deployed HIS system as well, however in the state of Himachal Pradesh, public health service has been using this centrally managed HIS registration portal which enhances the efficiency of healthcare operations, improves patient experience, and contributes to the overall quality of care within a hospital setting.

4.3.3 MPI reports portal (centrally managed)

The Master Patient Index (MPI) is a critical component of health information systems, particularly in electronic health records (EHR) or Hospital Information Systems (HIS). The MPI is a centralized repository that maintains a unique identifier for each patient and links that identifier to the patient's health records across various healthcare systems. This centralized MPI Reporting system has been deployed and maintained centrally at the public health service office to manage the overall patient-related information from all over the state healthcare systems connected to the public health services.

- This MPI report includes information on potential duplicate patient records. It helps healthcare organizations identify and resolve instances where multiple records exist for the same patient.
- MPI reports may focus on data quality, highlighting inconsistencies or errors in patient information. This could include missing information, incorrect spellings, or discrepancies in demographic details.
- This type of report assesses the accuracy of the MPI in matching and linking patient records. It helps in evaluating the effectiveness of the algorithms and processes used for patient identification.
- Reports may provide statistical data on the success rate of patient matching algorithms, including the number of matches, mismatches, and the overall accuracy of patient identification.
- These reports include metrics related to the management of patient identities, such as the frequency of updates, corrections, and additions to patient records.
- An audit trail report for the MPI provides a record of changes made to patient identifiers or records. This is crucial for maintaining data integrity and tracking user activity.
- Reports may be generated to ensure compliance with data governance policies and healthcare regulations related to patient information management.
- This type of report analyzes patient demographics to identify trends, patterns, and characteristics of the patient population served by the healthcare organization.
- If there are instances of potential duplicate records, a resolution report may document the actions taken to merge or resolve the duplicate entries.

MPI reports generated by a healthcare organization will depend on its information management policies, the technology used, and the specific challenges faced in
patient identification and record management. These reports play a crucial role in maintaining the accuracy and integrity of patient information within a healthcare system and the existing MPI Reporting system looks like as in the below screenshot.

Figure 17: MPI Reports System

4.3.4 Hardware and software requirements for OpenMRS

The hardware requirements for installing and setting up a Hospital Information System (HIS) can vary based on the specific software solution, the size of the hospital, and the expected workload. Here are general hardware requirements that are commonly considered when implementing a HIS in Hospitals at Shimla, Himachal Pradesh.

<table>
<thead>
<tr>
<th>Sr.No.</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Servers</strong></td>
</tr>
<tr>
<td>1</td>
<td>HP Proliant ML-330 with: Intel XEON Quad Core E5606 with Single Processor (upgradable to dual), 12MB L2 Cache / Intel 5500 series motherboard/ 4x4GB RAM PC2. 4x300GB SAS HOT Pluggable HD Drive/ RAID 5 implementation with 256MB Cache, 18.5&quot; TFT Analog Color Monitor/ 16x Max DVD+ RW drive/ 8MB VRAM display memory, Slots: 6 SATA Ports/ one serial, 6USB Ports, 4PCI slots, Certification for windows Server 2008 &amp; Enterprises Linux Edition, Without O.S. integrated Dual Ethernet Controller 10/100/1000 with full duplex, Server Management Software (with 6 years warranty)</td>
</tr>
<tr>
<td></td>
<td><strong>Desktop / Clients</strong></td>
</tr>
<tr>
<td>1</td>
<td>Computer System: - Intel Core Processor (i3, -i5)/ 4 GB DDR2 RAM expandable up to 8GB/ 500GB SATA HD Drive/ DVD-RW Drive/ integrated 10/100/1000MBPS LAN Card/ 18.5&quot; TFT Analog Monitor/ USB Keyboard/ Optical Mouse/ USB Ports/Lpt Port/ MacAfee Anti Virus-PL/ PL-Windows-7 prof. (O.S.) (with 6 year warranty)</td>
</tr>
<tr>
<td></td>
<td><strong>UPS</strong></td>
</tr>
<tr>
<td></td>
<td>For Desktops</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td>For Servers</td>
</tr>
</tbody>
</table>

### Network Switches

|   | DES-1016D 16 port 10/100M unmanaged standalone Switch (3 years carry-in warranty from the date of delivery) |

### Printers

<table>
<thead>
<tr>
<th></th>
<th>Dot Matrix Printer: - with 24 pins 136Col. 360CPS (12cpi) (3-year warranty on printer and Head except for ribbon &amp; plastic parts)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Laser printer with USB support</strong> (3-year warranty on printer and Head except for cartridge &amp; plastic parts). <strong>Print Resolution 600x600 DPI, Print Speed -12 PPM</strong></td>
</tr>
<tr>
<td></td>
<td><strong>Discretion to select printer rests with the state, as they deem fit.</strong></td>
</tr>
</tbody>
</table>

4.3.5 Software requirement for server and clients installation

To install OpenMRS, you need to ensure that your system meets certain software requirements. Below are the general software requirements for installing OpenMRS.

- Apache Tomcat 7
- Mysql 5.6 server
- Java Jdk7
- Mozilla Firefox
- Openmrs.war
- Birt.war
- Modules (.omod)
- Reports (.rptdesign)
- Adobe accrobat Pdf reader
- Ms office or open office
- Openmrs-runtime.properties file
- Addresshirchey.xml file

4.4 Teams Involved

To accomplish the objectives of my thesis, my supervisor at UiO Oslo, who is also the Founder and Head of HISP India, introduced me to the HISP India team. This team was actively engaged in the HIS OpenMRS Upgrade project for the public health service department in the Himachal Pradesh state. The HISP India team
operates from various locations within India, with the specific team handling the upgrade project situated in the cities of Bangalore, New Delhi, Mumbai, and Shimla. To facilitate direct collaboration, I traveled from Oslo to India to meet with the developers, testers, deployment engineers, project leaders, and project implementation coordinators. Over a period of two weeks, we engaged in face-to-face collaboration. Following this, I dedicated four months to hands-on work on the Reporting section of the new version of the Hospital Information System, incorporating the latest OpenMRS framework. This immersive experience provided valuable insights into the development and implementation processes, shedding light on both the successes and challenges faced by individual team members due to the distributed nature of development, testing, and implementation activities across various geographical locations.

Despite the physical distance, I maintained regular digital meetings with the team members, including fellow alumni researchers from Kenya. Additionally, I sought perspectives from friends and colleagues working in the IT field who have experience in distributed software development processes worldwide. Through digital media, I connected with them to gather opinions on the pros and cons of DSD.

This combination of on-site visits and digital collaborations allowed me to comprehensively understand the challenges and advantages associated with distributed development practices. It also provided a global perspective by involving individuals from different parts of the world who shared their experiences and insights on the topic. Below picture consists of my journey during my data collection and meeting different team members in Bengaluru, Delhi, and Shimla.
4.5 Challenges faced.

OpenMRS – Hospital is being a standalone implementation in all the sites since 2010 and upgrading this application to web-based application involved multiple resources spread across different part of India and me being in Norway, I encountered several challenges which I have categorized as different themes discussed below.

Communication and collaboration challenge

Interacting with the development team in India was bit challenging as the team was involved in multiple projects alongside the HIS OpenMRS upgrade like DHIS2 implementation, DHIS2 Mobile app development etc. The team expressed a need for clear specifications and ample time for delivering functionality. The demand for regular availability to provide feedback hindered uninterrupted development. According to the developers, the daily feedback to the project team across different locations was neither convenient nor deemed necessary. This situation underscores the importance of balancing the need for feedback with the development team's workflow to optimize productivity and meet project deadlines efficiently. Geographical dispersion presented obstacles to seamless collaboration and communication. Factors such as time zone variations, language differences, and the need for efficient
communication channels posed challenges in ensuring cohesive teamwork and effective information exchange.

**Technical expertise capacity challenge**

The upgrade project faced challenges with key resources being only partially available on any given day due to their simultaneous involvement in other projects. Additionally, work assignment and tracking were conducted through phone calls and emails. The absence of a dedicated project management tool to monitor the development progress emerged as a significant factor contributing to delays in delivery. This situation highlights the importance of efficient resource allocation, streamlined communication, and the adoption of appropriate project management tools to enhance collaboration and project tracking in a more organized manner. The diverse nature of the project exposed disparities in technical expertise among team members. Challenges arose in addressing gaps in knowledge related to web-based technologies, integration frameworks, and other essential technical aspects. Bridging these gaps became crucial for the successful execution of the project.

**Increasing needs for integration and operations**

From an implementation and operations perspective, the deployments in Shimla are primarily standalone, necessitating manual efforts and on-site availability for troubleshooting any issues that may arise. This indicates a decentralized and possibly resource-intensive approach to managing the deployments in Shimla. Consideration could be given to exploring more streamlined and automated processes, or the use of remote monitoring tools, to enhance efficiency and reduce the need for constant on-site presence for issue resolution.

### 4.6 Contribution during the case study

As a starting point, I started getting my hands dirty by working with the HISP India team who were working on the upgrade of OpenMRS Hospital software. So, I joined the team and started working on one of the modules called Reports Module, where in the User login with the registration clerk should be able to see patients registered and got the appointments to different departments\OPDs on the dashboard. My involvement in this project was part-time, conducted concurrently with writing my thesis and other relevant study tasks. The case study served as a practical opportunity for me to engage in an OpenMRS Hospital system upgrade project, offering valuable insights into the intricacies of such projects. Simultaneously, I aimed to enhance my programming skills through hands-on participation. This dual approach allowed me to apply theoretical knowledge from my studies to a real-world scenario and gain practical experience in the field. It's a commendable strategy for integrating academic learning with practical application.
• Gender wise patient report
• Patient category wise registration report. New and follow up patients report.
• Age wise registrations
• Total count of referral patients, online appointments, and ABHA creation
• User should have an option to select the date range or monthly or weekly data.
• User should be able to see data in graphical format and when user clicks on graph it should have an option to download the data in tabular format (Excel, PDF) as shown in the below screenshot.

![Figure 18: Report main page](image-url)
This case study not only provided me with practical experience but also served as a platform to apply theoretical knowledge from my studies to a real-world HIS OpenMRS system upgrade project. Simultaneously, I sought to enhance my programming skills through active participation. This dual approach allowed me to bridge the gap between academic learning and practical application, offering valuable insights into the complexities of system upgrade projects in the field of Health Information Systems.
After my release from the project work with the team, significant development activities and changes have been implemented in the system. Unfortunately, I currently do not have access to this updated version of the system while writing my thesis.

4.7 Observation and discussion

With my case study on the upgrade of OpenMRS – Hospital has provided valuable insights into the challenges faced by DSD. The identified challenges can be categorized into several themes, shedding light on critical areas that require attention for enhancing the effectiveness of distributed development. These themes include:

**Communication and collaboration:** Geographical dispersion and diverse team locations have highlighted challenges in effective communication and collaboration. Addressing these challenges is crucial for fostering cohesive teamwork and ensuring seamless information exchange.

**Technical expertise capacity:** Disparities in technical expertise among team members emerged as a significant challenge. Bridging these gaps in knowledge related to web-based technologies and integration frameworks is essential for the successful execution of the DSD project.

**Need for appropriate collaborative software tools:** The complexities of a distributed development environment call for the implementation of appropriate collaborative software tools. Identifying and integrating tools that enhance communication, project management, and collaborative workflows is essential for overcoming the challenges posed by geographical dispersion.

**Knowledge acquisition and management issues:** Challenges related to knowledge acquisition and management have been identified as critical. Effectively capturing, sharing, and managing knowledge within a distributed team is essential for maintaining a cohesive understanding of the project and its requirements.

**Outcome of my Interviews:** During my face-to-face interview with a senior Java developer in Bengaluru, who was actively involved in the development of Java APIs for OpenMRS, several insights were shared. According to the developer, distributed software development presents significant challenges related to distance, language barriers, software and project management tools, and the high cost associated with hiring skilled developers or engineers. He emphasized that overcoming these challenges is crucial for the successful implementation of distributed software development.

The interviewee also expressed the belief that the integration of effective project management and application monitoring tools could greatly enhance productivity and
accelerate the development process. Specifically, for the HIS OpenMRS system, the developer supported the idea of introducing a mobile app. According to him, incorporating a mobile app into the OpenMRS implementation would be a noteworthy initiative, positioning it as a competitive software product, particularly in comparison to the CDAC HIS software. This suggestion underscores the importance of continuous innovation and adaptation to stay competitive in the dynamic field of healthcare information systems.

Continuing my journey, I proceeded to meet the HISP India team in Delhi, engaging with a senior DHIS2 and OpenMRS architect, as well as a front-end developer. During these meetings, I gained crucial insights into the HIS OpenMRS application's dynamics, particularly in comparison to a competing application developed by CDAC.

While the CDAC application excels in the Drugs and Pharmacy-related module, the HIS OpenMRS distinguishes itself with a broader range of functionalities, attracting health workers to choose it over the CDAC alternative.

The diligent efforts of the HISP India development team, coupled with persuasive discussions with the government of Himachal Pradesh, resulted in obtaining necessary certifications. These certifications allowed the application's deployment in numerous health centers.

A notable challenge discussed was the limited presence of the application in various hospitals. The proposed solution is to expand implementation to all healthcare service centers, enabling the collection of more usage data and fostering the vision of a centralized software accessible to all state hospitals.

The senior architect recommended transitioning to a cloud-based infrastructure, emphasizing its advantages over on-premises solutions, such as scalability and flexibility. Additionally, the importance of monitoring, both in terms of infrastructure and application, was highlighted as pivotal for achieving optimal performance.

Subsequently, I journeyed to Shimla to meet the HIS OpenMRS implementation team at the HISP India Shimla office. During this visit, I engaged with a group of testers and senior deployment engineers who provided a detailed explanation of their installation procedures. This involves personally visiting hospitals after acquiring the required hardware and software for the hospital premises.

Following the installation process and ensuring the application is operational, the same team is responsible for the ongoing operations and maintenance of the application. In case of any issues, hospital staff contacts the engineers, who then go on-site to check and troubleshoot the problems. The consensus among developers and engineers in Shimla also aligns with the recommendation for a cloud-based
application deployment, emphasizing that such an approach, coupled with necessary monitoring tools, would enhance efficiency, and expedite issue resolution.

Interviews, typically conducted on a one-to-one basis, can also involve a group of respondents, known as focus group discussions (Oates, 2005). In addition to utilizing semi-structured interviews, this study incorporated data collection through focus group discussions. Within the HIS OpenMRS implementation project, meetings and discussions were held with health program coordinators and other hospital staff.

The focus group discussions provided valuable insights that complemented the data obtained from semi-structured interviews. These discussions aimed at addressing challenges and conflicts associated with HIS OpenMRS, involving various stakeholders. While the interviews weren't directly linked to the HIS implementation, they shed light on the routine challenges encountered with the application. Moreover, they contributed to shaping my understanding of HISP operations in other developing countries, offering a platform for sharing and learning from diverse experiences.

**Online survey results:** The online survey with 24 respondents on distributed software development revealed a high level of familiarity among respondents, with approximately 96% having direct involvement in various stages of the distributed software development lifecycle. The survey, conducted to gauge perceptions and experiences in this domain, provides valuable insights into the prevalence and engagement of individuals in distributed software development practices. as shown in the below charts.

![How familiar are you with distributed software development?](chart.png)
During the survey, participants were tasked with outlining the challenges they encountered in their involvement with Distributed Software Development (DSD). In addition to detailing the nature of these challenges, respondents were also prompted to articulate the solutions they had devised to overcome the identified problems. Furthermore, they were asked to evaluate the success of these solutions, providing a comprehensive understanding of the strategies and effectiveness in addressing the hurdles associated with DSD. As a result, I have received below challenges listed out by the respondents, many have restated the communication and co-ordination as the main issue with other challenging factors.

- Lack of coordination and being synchronized.
- Lack of clear statement of requirements.
- Communication is one of the foremost challenges when doing distributed software development.
- With teams in different physical locations social capital is difficult to build up affecting team motivation.
- Conflicts within the teams.
- Clearing the security and compliance requirements.
- Time zone for meetings, Skillset, Lack of test case automation.
- Distribution, Vulnerability management, Traceability.
- Reliability and Fault Tolerance are tricky to manage.
- Collaboration and Security.
- Poor communication can be a challenge sometimes, in a distributed space, it is important that there is clear communication about priorities and tracking progress.
- Accessing, authorizing.
- Communication, Collaboration, Cultural Differences, Time Zones, Security and Data Privacy, Technical Challenges, Project Visibility, Knowledge Sharing,

- Time zones, feeling of one team, understanding of requirements.
- Communication.
- Clarity of the requirements to all the stakeholders.
- Limited or no dedicated Testing team.

Participants response for the strategies used to overcome these challenges are.

- Introducing Release Management tools like Spinnaker etc.
- For software development use teams/zoom and build a devops culture. Use of Git and build effective pipelines.
- Yes, Implemented Jira to track the progress and issues.
- More show and tell sessions to educate, Daily stand-up meetings.
- Yes, Using of Deployment center tool base on customer
- Some sessions on working culture, having Effective tool for communication.
- Having video call at one time like stand ups where everyone is present, Slack groups.
- Automatic standups via application where they can update the status, blockers and other important things.
- Business flow monitoring to quickly identify issues.
- Tools like Splunk, Grafana, mix panel and power bi help in discovering issues in distributed systems.
- Yes, training the concerned resource and documenting the training for future reference.

This case study highlights the multifaceted challenges inherent in Distributed Software Development, emphasizing the need for research and solutions in key areas such as communication, technical expertise, collaborative tools, and knowledge management. Addressing these challenges will contribute to making distributed development more effective and streamlined. As per my observation, the challenges inherent in distributed development often result in significant drawbacks, particularly in terms of slowing down multi-site work. While the production costs may be comparatively low in DSD, there is a notable increase in coordination costs. The intricacies of managing teams across different locations, time zones, and cultural contexts introduce complexities that can impede the efficiency of collaborative efforts. The cost-benefit trade-offs in distributed development have become a focal point of interest for both researchers and practitioners. Numerous studies have been conducted to explore and understand the dynamics of coordination in distributed software teams and geographically dispersed teams in general. These research endeavors aim to unravel effective strategies for mitigating the challenges posed by
coordination costs and optimizing the overall efficiency of distributed development practices.

5. Case analysis

Data collected through my fieldwork through hands-on development, interviews and group discussions, and online survey were analyzed using qualitative methods. The research results serve to highlight key challenges and clarify specific areas for improvement in distributed software development processes. Furthermore, the findings suggest relevant work practices that have been tested in an industrial environment, providing actionable insights to enhance and optimize the practices associated with distributed software development. As part of my observation and data gathering from HISP India team, Software professionals across different organizations and global locations, following are the key challenges in a distributed software development lifecycle that are observed by me and confirmed by many other software professionals. After my data collection as discussed in previous sections, I started my analysis of the gathered data.

5.2 Answers to the research questions

Based on my analysis, I will answer the research questions posed in my thesis

I. What are the key challenges/impediments of DSD?

As an outcome of this study, main challenges that were gathered as part of my data collection are as follows which is very well syncing with the theoretical challenges that were discussed in my Introduction chapter. I have also observed the same areas that are discussed under the challenges section of the article published by Hindawi Publishing Corporation.

- Communication gaps between the team members: The software life cycle necessitates extensive communication among development team members, involving the exchange of substantial information through various tools and formats, often without adhering to standardized communication practices. This lack of standardization leads to misunderstandings and prolonged response times. The challenges, compounded by the intricate infrastructure and the dynamic nature of personal networks, contribute to a decline in the frequency and quality of communication, directly impacting overall productivity. Recently I received the below cartoon from my usual communication channel at work which is the best illustration of usual communication problems in software projects. This difficulty to capture the customer needs and correctly translate them in the form of requirements, design and build up an application with the working code especially involving the development team across the globe is very clearly shown here.
• **Weak co-ordination with cross workstreams during development phase:** Coordination in multisite developments becomes notably challenging in terms of articulation work. Issues stemming from communication problems, a lack of group awareness, and the inherent complexity of the organization influence the structuring and management of work. Multisite communication and coordination demand the involvement of a larger number of participants, leading to potential delays. Implementing significant changes across multiple sites requires more time, involving numerous individuals. Consequently, more progress reports, project reviews, conference calls, and regular meetings are essential to identify and address issues in a timely manner.

• **Cultural differences of employees spread across the globe:** I encountered this challenge, while working with the team members who are spread across the country and are in different work culture, where if one says about a functionality change and need to meet up for discussion, the other person in different location interpret in different manner. As highlighted in the report\(^{29}\) 60\% of all global IT companies will outsource some software development and in that two-thirds of all software projects end in partial or total failure\(^ {30} \), as the cultural differences between the project stakeholders, play a very important role in the final deliveries. Recognizing cultural differences is a fundamental and initial step in preventing misinterpretation and other errors within a software development project. It's crucial to acknowledge that offshore outsourcing involves individuals with diverse backgrounds, languages, cultures, and values. Therefore, overcoming cultural differences is a vital and necessary step for the success of offshoring software development projects.
• **Impact on project deliveries due to different time zone:** I was working with the team on part time basis, so many other developers in the team as well. This made the individuals unavailability all at once, was a key drawback in the delivery of the assigned tasks to each other. Once, when the task was assigned by the coordinator to me and was asked to complete in 3 days, but due to other team members unavailability, my code merge took long time to get reviewed and merged to the main branch. Working with teams in different time zones presents challenges in synchronizing schedules and coordinating tasks. The potential result is delays in project delivery, which can significantly impact the overall success of the project. Managing time zone differences effectively becomes crucial to mitigate these challenges and ensure smooth collaboration and timely project completion. During my online survey, when I asked the question about How does they handle time zone differences when collaborating with team members in different location, responders have chosen 50% of them will adjust their working hours, which is a challenging decision to make adjusting their work life balance by doing so. Below is the result captured from the online survey summary.

How do you handle time zone differences when collaborating with team members in different locations?
24 responses

![Chart showing survey results]

- Adjust Working Hours: 50%
- Use asynchronous communication: 33.3%
- Rotate Meeting times: 6.6%
- Use async communication and whenever necessary have a meeting to talk with the team to convey the requirements: 16.6%
- Having a meeting late in the day, which was early in the morning for the other team: 6.6%

• **Knowledge management and sharing Issue:** Though this OpenMRS Hospital application was deployed back in 2010, there was no proper documentation on the initial implementation, versions used, and issues and fixes done during these many years were not properly documented in any manner for the future workforce to have it as knowledge base. In geographically distributed teams, the sharing of explicit knowledge is crucial for which documentation plays a vital role. However, in distributed development, there is often a deficiency in proper documentation. Many organizations nowadays adopt agile approaches, which, while emphasizing sharing tacit knowledge, may not support extensive documentation. This results in much of the product knowledge being embedded in source code,
test files, and outdated documentation, posing a challenge for dispersed team members who rely on documentation for understanding the product.

In addition to documentation challenges, there is an issue of knowledge vaporization. Both local and remote team members need to communicate effectively to collaborate on the same project, but much of the knowledge is dispersed across electronic media, such as chat logs, making retrieval a non-trivial task. Addressing these challenges is crucial for enhancing collaboration and knowledge sharing in distributed development teams.

- **Lack of software collaboration tools:** It was a major difference I experienced, while working with the team during my case study, where no collaboration tools were used during any SDLC process. In today's DevOps landscape, teams operate in hybrid environments, spanning offices, homes, and various geographical locations. This remote and distributed model introduces complexities to the development process. Challenges arise in sharing knowledge and communicating in real-time, especially when teammates are in different time zones. Additionally, the peer review process for code becomes more intricate when the traditional proximity of colleagues is absent.
These challenges underscore the importance of software development tools that facilitate and enhance collaboration. Solutions that address remote communication, real-time knowledge sharing, and streamlined peer review processes are essential for the efficiency and success of DevOps teams in hybrid and distributed work environments.

- **Not having clear process-oriented project planning:** In distributed environments, high organizational complexity poses challenges in various aspects such as scheduling, task assignment, and cost estimation. These challenges are exacerbated by factors like volatile requirements, changing specifications, cultural diversity, and a lack of informal communication. Managers are tasked with overseeing the development process comprehensively, striving to enhance it during enactment while mitigating factors that may impede productivity. This involves considering the potential impact of diverse cultures, identifying interrelated tasks, and minimizing dependencies among distributed groups. Effectively navigating these complexities is crucial for successful development in distributed settings.

- **Standalone application deployments make things very fault intolerance:** My experience on DSD and SDLC was far matured than the implementation, that I saw when I started working on this project. Running a software on a single instance infrastructure located at the hospital premises, where no technical staffs are available makes the implementation too error prone. While working on DSD environment, where the team members are located across the globe, it becomes very difficult to deploy and maintain the software application in the standalone infrastructure such as in individual hospitals, offices etc. This unlatches the main theme of distributed software development concept and brings out many challenges in operating such an implementation.

- **Holding up on work force is an expensive thing:** My interaction with developers revealed that, team members tend to switch jobs for the better financial perks and finding a right person with technical knowledge about OpenMRS was a bigger challenge. Though the main advantage of adopting to the distributed software development concept is to open for the global workforce, sometimes while addressing the other challenges listed above, it will become bit an issue of having a workforce, who are experts in the software tools that project is working on and keeping such resource in the team for long time due to the job attrition rate. As reported in the entrepreneur.com here are the most common reasons for the engineers to quit their job in any organization.
II. How do we address these challenges to improve the DSD?

With my study on exploring the challenges/impediments of DSD, I have got quite a bit hold of the subject and from the earlier works from different researchers referenced in my study, I have understood that recognizing the challenges inherent in DSD is crucial to mitigating the risk of development failure and optimizing the chances for success. The results of this study contribute to identifying and elucidating key areas for improvement in distributed software development processes. Moreover, the findings suggest relevant work practices that have been tested in an industrial environment\textsuperscript{32}, providing valuable insights to enhance the effectiveness and success of DSD initiatives. After establishing the key challenges listed in the previous section, it’s now the time to explore the improvement measures for those problems and challenges faced in distributed software development process.

- **Addressing the communication gaps between the team members by introducing the communication and collaboration tools:** During my case study, communication was basically on zoom and based on demand from the team members. To bridge the communication gap in distributed environments, it is imperative to support methodologies and processes with collaborative tools. These tools serve as a mechanism to mitigate ambiguity and compensate for the absence of face-to-face meetings without compromising the quality of results. Recognizing the importance of user-friendly tools is essential, and integrating collaborative tools and agents becomes a strategic approach to enhance knowledge integration. It is crucial to discuss the need for these user-friendly tools and delve into collaborative processes, including

![Figure 21: Job attrition trend from entrepreneur.com](image)
techniques such as conference calls and email, to foster effective communication in distributed teams. Introducing a practice of daily standup of 15 mins, would make things clarified better than assuming things and working on the task. Introducing the collaborative tools like Jira, Teams and proper version control would help the project teams to overcome the communication hurdle. It has been recommended in many industrial practices, the results from my online survey and with 20 years of my professional experience in the software industry.

![Figure 22: Communication and Collaboration tools](image)

- **Removing the co-ordination issue with cross workstreams during development phase by involving a project manager and the necessary tracking tools:** Our team was spread across many geographical locations, however due to the communication gaps, developers being hesitant or ignorant, interaction use to halt for longer duration. To address this issue, an increased frequency of progress reports, project reviews, conference calls, and regular meetings for corrective action is necessary. This approach helps minimize task dependencies with other locations. Collaborative tools play a crucial role in supporting analysis, design, and development, enabling the monitoring of activities and the management of dependencies. These tools facilitate notifications and the implementation of corrective measures, fostering a more effective and coordinated approach to distributed development projects. Also, I recommend, a project manager, who plans the project time lines, assign the tasks, and enable co-ordination between team members would make the team more productive.

- **Acting up on cultural difference of employees spread across the globe:** My online survey respondents highlighted the significant challenge posed by
sociocultural distance, also as discussed in this article, emphasizing the difficulty of establishing mutual understanding among individuals with diverse backgrounds. The limitations extend beyond vocabulary, encompassing varying cultural interpretations of communication. While there may be a general proficiency in English, subtle issues such as political or religious values can lead to misunderstandings and conflicts within projects. Companies strive to address this challenge by actively promoting both informal and formal knowledge sharing among project participants. Ultimately, the effectiveness of overcoming sociocultural differences hinges on the capacity and interest of individuals in understanding one another. As per my observation and experience, improving the project level documentation and knowledge sharing session, would assess this situation for some extent. Of course the topic in question cannot be addressed in any one approach, but a project manager can assess the situation and take necessary action as and when it is necessary.

- **Allowing flexibility of project delivery to address different time zone problem**: As most of the respondent in our online survey had replied that adjusting working hours to accommodate to interact with all the global team members, it's a good gesture that, even the organization to join hands in updating the working hours policy to flexible way, so that employees can balance their work life activities fairly.

  ![Chart](chart.png)

  - Adjust Working Hours
  - Use asynchronous communication
  - Rotate Meeting times
  - Use async communication and whenever necessary have a meeting to talk with the team to convey the requirements
  - Having a meeting late in the day, which was early in the morning for the other team

- **Introducing the document management tool to maintain the project knowledge to share with the team members**: I have come across, tools like SharePoint, confluence and many more even with the opensource world to create, store and share the knowledge within the team in secure manner. In distributed environments, effective knowledge sharing necessitates the establishment of a product/process repository that concentrates on well-understood functionality. This repository should link content from diverse sources, including email and online discussions, and share metadata information across various tools. This approach ensures a centralized and
accessible knowledge base, facilitating collaboration and information retrieval in distributed teams. I recommend, the implementation of such a tool which improves the DSD projects knowledge base for longer period and address the knowledge vaporization issue.

- **Invest in tools for better communication and collaboration:** To cope with today's DevOps landscape, teams operate in hybrid environments, spanning offices, homes, and various geographical locations. So, including tools such as slack, teams, skype of better communication and development life cycle management tools like Jira, Jenkins, ansible, terraform and few log analytics and application monitoring tools which would make DSD a better concept.

- **Following the Project management Principles:** As stated in the article, project management emerges as a significant challenge, with 59% of respondents identifying it as the fourth most significant problem area. Addressing project management challenges is more complex in distributed environments compared to centralized development settings. Notably, in distributed development, a substantial amount of effort is required for upfront planning and follow-up activities to successfully manage a project. Failure to recognize this need at the project’s outset can lead to uncertainty, misunderstandings, and management problems later. Managing a globally distributed project demands a diverse skill set and knowledge, including cultural awareness, strong communication skills, technical competence, and particularly adept project management capabilities.

- **Avoiding Standalone application deployments and opting for Centralized or cloud-based deployments would make system more fault tolerance:** This was the challenge specific to my case study and would like to add my opinion and suggestion to overcome this challenge. Choosing centralized or cloud-based deployments over standalone application deployments presents a strategic move with several benefits, especially in terms of enhancing the fault tolerance of the system. I did experience both the standalone situation by working installing the application on my local laptop and installing it on a cloud environment. Which the later option made my easier in accessing the application smoothly.
  
  o **Scalability:** Centralized or cloud-based deployment models often come with scalable infrastructure, allowing the system to adapt to changing demands seamlessly. This scalability ensures that the system can handle increased loads without compromising its performance or reliability. However, what is seamless by having it on cloud is its quick and efficient to scale up and down the system with no long waiting time for the procurement of infrastructure as it is with standalone environment.
Accessibility: Centralized or cloud-based systems provide accessibility from various locations and devices. This accessibility improves the overall user experience and ensures that the system remains available even in the face of localized failures.

Reliability: Cloud services and centralized deployments often leverage redundant and geographically distributed servers. This redundancy ensures that if one server or location experiences issues, the system can continue functioning without significant disruptions. This reliability is crucial for fault tolerance.

Resource Efficiency: Cloud-based deployments enable efficient resource utilization. The dynamic allocation of resources based on demand ensures optimal performance, reducing the risk of resource-related failures.

Data Backups: Centralized or cloud-based deployments often come with robust data backup mechanisms. Regular backups and distributed storage systems contribute to data resilience, mitigating the impact of data-related faults.

Fault Isolation: Centralized or cloud-based architectures often include mechanisms for fault isolation. If a fault occurs in one part of the system, it can be isolated without affecting the entire system's functionality.

Automatic Updates and Maintenance: Cloud-based platforms often handle updates and maintenance tasks automatically. This ensures that the system is running on the latest software versions with patched vulnerabilities, reducing the risk of security-related faults.

Cost Efficiency: While there might be initial costs associated with migrating to centralized or cloud-based deployments, the overall cost efficiency can be higher. Pay-as-you-go models and shared infrastructure contribute to cost savings over time.

In summary, transitioning from standalone application deployments to centralized or cloud-based deployments is a strategic move that not only improves fault tolerance but also enhances scalability, accessibility, reliability, resource efficiency, data backups, fault isolation, and overall cost efficiency. It aligns with modern best practices in system architecture and management.
• **Job attrition issue:** My observation regarding job attrition trends aligns with the dynamic nature of the contemporary job market, especially in the digital era where opportunities abound. While addressing the challenges associated with DSD can indeed contribute to a more balanced work-life equation, it’s crucial to acknowledge that job attrition is a multifaceted phenomenon. I have seen the job attrition trend never be with one reason. In today’s digital world, more and more job opportunities pulling out the key experts from job to job. Addressing most of the above challenges list would make team members to have a good work life balance and would avoid quitting for better option. Of course, this challenge not only rely on the distributed software development aspect, but other personal aspects such as family condition, financial need, self-development, and growth aspirations etc. In this context, organizations that prioritize addressing both the professional and personal needs of their team members are likely to foster a more supportive and satisfying work environment. By offering opportunities for skill development, career growth, and maintaining open channels of communication, companies can create an environment that encourages loyalty and reduces the likelihood of talent attrition.

### 6. Conclusion

According to this thesis work, it underscores the importance of recognizing the challenges inherent in DSD as a crucial step toward minimizing the risk of development failure and maximizing the chances for success. The outcomes of this research not only identify these challenges but also shed light on specific target areas for improvement within DSD processes.

Moreover, my research contributes valuable insights by suggesting and clarifying appropriate work practices. These practices, tested in an industrial environment, provide practical and actionable recommendations for enhancing the effectiveness of DSD as discussed in chapter 5. In essence, this study serves as a guide for practitioners and organizations engaged in DSD, offering targeted strategies to mitigate challenges and optimize processes. By acknowledging and addressing these challenges head-on, my research contributes to fostering a more resilient and successful environment for DSD projects.

From the details of this study, we should understand that the complexity of software development processes is further compounded when distributed over multiple sites. The distributed nature introduces additional layers of intricacy, requiring effective management of knowledge and communication across geographically dispersed teams. This complexity would require a very well manageable communication and
collaboration tools and the knowledge management process in place as discussed in detail in chapter 5.

I have organized the previously identified challenges into few categories: communication and collaboration challenges, knowledge vaporization challenge, technical expertise capacity challenge, project management challenges and the lack of development and collaboration tools challenge. In addressing my research questions, I delved into detailed discussions about best practices to overcome these challenges. However, considering the vastness of DSD as a research field, the presented work is not a conclusive and final set of recommended best practices. It serves as a starting point, acknowledging the need for ongoing evolution and further studies to establish more permanent solutions for these challenges. This work lays the foundation for future endeavors in DSD research, and I intend to continue my research in this area, contributing to the advancement of knowledge and practices within the field of DSD.

6.1 Future Work

The research in this thesis has its limitations and can be complemented with further research, based on the observations and findings on the challenges identified as part of my study, if time permits and I get an opportunity to further study, I would like to propose following listed work for further research and improvement work with respect to my case study.

6.1.1 Implementing an open-source project and task management tool.

Open-source project and task management tools provide organizations with flexible and customizable solutions for planning, organizing, and tracking their work. Here are some popular open-source tools in this category:

- OpenProject: Open-source project management software.
- Taiga: Offers agile project management features.
- Kanboard: Simple and visual task board.
- Tackboards: A PHP-based Kanban board.

Various tools available in the market serve diverse needs, ranging from simple task management to comprehensive project planning and collaboration. The selection of these tools is contingent upon the specific requirements and preferences of the organization or development team. In the subsequent phase of further investigation, the focus will be on addressing specific challenges highlighted in my thesis, specifically targeting communication and collaboration issues along with project management challenges. This strategic approach aims to leverage appropriate tools
to enhance communication, streamline collaboration, and improve overall project management processes.

6.1.2 Implementing Mobile app for OpenMRS HISs
During my field study, while discussing different improvement options with developers, implementation teams and the hospital staffs who are using the HIS OpenMRS application are very much interested and eager to have a mobile app to do their job on mobile phone. This opens an idea and makes me curious to look at the possibility of creating a mobile app as we have one for DHIS2 at least for few functionalities like user registration, appointment booking and reporting functionalities. Initiation of this project in DSD environment, will allow me to further investigate the depth of the challenges and come up with much more specific solution to the problem.
7. References

6. What Is SDLC (Software Development Lifecycle)
8. IT Services Outsourcing Market
11. https://wiki.openmrs.org/display/docs/Introduction+to+OpenMRS
12. Distance Matters by Gary M. Olson and Judith S. Olson University of Michigan
17. https://openmrs.org/
18. https://www.who.int/health-topics/universal-health-coverage#tab=tab_1
26. https://www.duo.uio.no/handle/10852/9530
27. Using thematic analysis in psychology Virginia Braun & Victoria Clarke https://www.tandfonline.com/doi/abs/10.1191/1478088706QP0630A
29. the Accelerance Global Software Outsourcing report

81
36. UiO, Hisp, OpenMRS
38. https://www.precedenceresearch.com/it-services-outsourcing-market
ANNEXURE A: Online Survey Questionnaire

Distributed Software Development and Distribution Challenges and Improvements - Questionaire

This Questionnaire is to collect valuable data and insights from individuals or teams involved in distributed software development and distribution, which helps better understand the current landscape, challenges, and potential areas for improvement in this field. The gathered information can inform decision-making, strategy development, and process optimization in software development projects.
https://www.techopedia.com/definition/28959/distributed-development-software-development

I understand that my personal information, including my name and contact details, will be kept * confidential and will not be shared with third parties. I consent to the use of my responses for research purposes while maintaining my anonymity.

☐ I consent to the privacy and confidentiality terms.

Name

Short-answer text

Email

Short-answer text

Organization/Company Name

Short-answer text
Role
Short-answer text

Distributed Software Development and Distribution
This section aims to understand the respondent's involvement in distributed software development projects and their level of experience in this context.

How familiar are you with distributed software development?

- Very Familiar
- Familiar
- Somewhat Familiar
- Not Familiar

Have you been involved in distributed software development projects? (Check all that apply)

- Yes, as a developer
- Yes, as a Project Manager
- Yes, as a Tester
- Yes, as a Documentation Writer
- No

What are the main challenges you've encountered in distributed software development?

Short-answer text
What collaboration tools and technologies do you use in distributed software development projects?

- Email
- Video Conferencing (e.g., Zoom, Teams)
- Chat/Messaging Apps (e.g., Slack, Teams, Etc)
- Version Control Systems (e.g., Git)
- Project Management Tools (e.g., Jira)
- Other...

How do you handle time zone differences when collaborating with team members in different locations?

- Adjust Working Hours
- Use asynchronous communication
- Rotate Meeting times
- Other...

What challenges have you encountered in distributing software to a global audience?

- Network Latency
- Compliance with local regulations
- Language and Localization
- Platform specific issues (e.g., OS compatible issues)
- Other...
<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>How do you distribute software to end-users or clients? (Check all that apply)</td>
<td>Direct downloads from a website, Through app stores (e.g., Apple App Store, Google Play), Physical media (e.g., DVDs), Cloud-based distribution (e.g., AWS, Azure), Other (please specify)</td>
</tr>
<tr>
<td>What are the key considerations when deciding on a distribution method for your software?</td>
<td>Short-answer text</td>
</tr>
<tr>
<td>How do you handle software updates and patches for distributed software?</td>
<td>Long-answer text</td>
</tr>
<tr>
<td>Have you incorporated open-source software into your projects?</td>
<td>Yes, No, I am Not Sure</td>
</tr>
<tr>
<td>If yes, how do you ensure compliance with open-source licenses?</td>
<td>Long-answer text</td>
</tr>
<tr>
<td>What do you think are the challenges in distributed software development and distribution?</td>
<td>Long-answer text</td>
</tr>
</tbody>
</table>
Have you implemented any specific strategies or tools to address these challenges? Please describe.

Long-answer text

How do you manage project documentation and knowledge sharing in distributed teams?

Short-answer text

What do you think are the emerging trends in distributed software development and distribution?

Short-answer text

Please provide any additional comments, insights, or suggestions related to distributed software development and distribution.

Long-answer text

ANNEXURE B: Online Survey Questionnaire response from 24 respondents
ANNEXURE C: Online Survey results Summary

24 responses

I understand that my personal information, including my name and contact details, will be kept confidential and will not be shared with third parties. I consent to the use of my responses for research purposes while maintaining my anonymity.

24 responses

100%

I consent to the privacy and confidentiality terms.

Name
24 responses

[Names redacted]

View in Sheets

Copy
Email
23 responses
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com
@<redacted>gmail.com

Organization/Company Name
19 responses

<table>
<thead>
<tr>
<th>Company Name</th>
<th>Response Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altera</td>
<td>2 (100%)</td>
</tr>
<tr>
<td>DNB</td>
<td>1 (5.3%)</td>
</tr>
<tr>
<td>GEC, Hassan</td>
<td>1 (5.3%)</td>
</tr>
<tr>
<td>Infosys Limited</td>
<td>1 (5.3%)</td>
</tr>
<tr>
<td>Kühne Nagel GmbH</td>
<td>2 (100%)</td>
</tr>
<tr>
<td>Nghi</td>
<td>1 (5.3%)</td>
</tr>
<tr>
<td>Tata Consultancy Services</td>
<td>1 (5.3%)</td>
</tr>
<tr>
<td>UHG</td>
<td>1 (5.3%)</td>
</tr>
</tbody>
</table>
Role
22 responses

Cloud Architect
Faculty
Software Engineering - LMTS
Devops Engineer
Devops
Head of technical platforms
Design Engineer
Senior Tech Arch
Principal Architect

Distributed Software Development and Distribution

How familiar are you with distributed software development?
24 responses

- Very Familiar: 41.7%
- Familiar: 29.2%
- Somewhat Familiar: 25%
- Not Familiar: 4.1%
Have you been involved in distributed software development projects? (Check all that apply)

23 responses

- Yes, as a developer: 15 (65.2%)
- Yes, as a Project Manager: 7 (30.4%)
- Yes, as a Tester: 2 (8.7%)
- Yes, as a Documentation Writer: 2 (8.7%)
- No: 2 (8.7%)

What are the main challenges you've encountered in distributed software development?

19 responses

- Lack of coordination
- Lack of clear statement of requirements
- Communication is one of the foremost challenges when doing distributed software development. 2) With teams in different physical locations social capital is difficult to build up affecting team motivation
- Conflict
- Being synchronized
- Clearing the security and compliance requirements
- Timezone for meetings, Skillset, Lack of test case automation
- Distribution, Vulnerability management, Traceability
- Reliability and Fault Tolerance are tricky to manage.
What collaboration tools and technologies do you use in distributed software development projects?

24 responses

- Email: 12 (50%)
- Video Conferencing (e.g., Zoom): 16 (66.7%)
- Chat/Messaging Apps (e.g., Slack): 18 (75%)
- Version Control Systems (e.g., Git): 18 (75%)
- Project Management Tools (e.g., Jira, Asana): 16 (66.7%)
- Jenkins, Confluence, puppet: 1 (4.2%)
- Build systems, Observability tools: 1 (4.2%)
- SharePoint, Azure Devops, Asana: 1 (4.2%)

How do you handle time zone differences when collaborating with team members in different locations?

24 responses

- Adjust Working Hours: 33.3%
- Use asynchronous communication: 50%
- Rotate Meeting times: 16.7%
- Having a meeting late in the day, which was early in the morning for the other team: 16.7%
- Use async communication and whenever necessary have a meeting to talk with the team to convey the requirements: 16.7%

What challenges have you encountered in distributing software to a global audience?

24 responses

- Network Latency: 7 (20.2%)
- Compliance with local regulations: 15 (62.5%)
- Language and Localization: 11 (45.8%)
- Platform specific issues (e.g., OS compatible issues): 7 (20.2%)
- Time zones: 1 (4.2%)
- License: 1 (4.2%)
How do you distribute software to end-users or clients? (Check all that apply)

- Direct downloads from a website: 7 (30.4%)
- Through app stores (e.g., Apple App Store, Google Play): 7 (30.4%)
- Physical media (e.g., DVD): 0 (0%)
- Cloud-based distribution (e.g., AWS, Azure): 18 (78.3%)
- Other (please specify): 3 (13%)

What are the key considerations when deciding on a distribution method for your software?

- Latency, Failover, Zero downtime
- The type of product, the target market, the cost and availability of the channel, and the level of control and support that the producer wants to have over the product.
- Trust, Availability
- Only the right people should get access to the software. Principle of least privilege
- Make sure the team or each individual is working towards the same goal.
- Ease of access, access to and being informed about updates, and what they include and more
- Micro services
- Security, avoiding piracy
- Ease of use to end user
How do you handle software updates and patches for distributed software?

17 responses

- By designing stateless microservice based architecture to avoid updates & patches
- Use patch management software to automate the process.
- Replicated updates and switching-over
- Microsoft Intune
- We need to do those with zero down time.
- It depends on the project, but keeping backwards compatibility, using versioned APIs, and more.
- Patch updates in staggered manner
- Make updates available in git and upgrades to happen with few commands
- Golden images (VM, Containers)

Have you incorporated open-source software into your projects?

24 responses

- Yes: 13 (54.2%)
- No: 10 (41.7%)
- I am Not Sure: 2 (8.3%)
If yes, how do you ensure compliance with open-source licenses?

13 responses

- Read the license terms carefully and adhere
- Very rarely thought about it
- It was handled by the vendors open source handling
- Based on licensing.. add the distribution license and also internal osrb process (open source review board)
- IP checks and legal
- only use company approved open source software
- Continuous integration tools for Open Source License Compliance
- We only use open source software which have permissive licences like MIT, Apache etc.
- Going through legal and license agreements

What do you think are the challenges in distributed software development and distribution?

20 responses

- Lack of co-ordination with cross workstreams during development phase. Dependencies which would delay the distribution
- Coordination and communication become more difficult as the software components are sourced from different places, thus affecting project organization, project control, and product quality.
- Proper identification and elimination of bottlenecks
- Communication for software development. Licensing for software distribution
- Communicate the tasks internally.
- There are benefits in having a team in the same place, as lots of interaction can happen when people sit together. So being structured, but also allowing for the unstructured is difficult.
- Orchestration of micro services
- Catching up with competition
### Have you implemented any specific strategies or tools to address these challenges? Please describe.

17 responses

<table>
<thead>
<tr>
<th>Strategies</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release Management tools like Spinnaker etc.</td>
<td></td>
</tr>
<tr>
<td>No</td>
<td></td>
</tr>
<tr>
<td>For software development use teams /zoom and build a devops culture. Use of Git and build effective pipelines</td>
<td></td>
</tr>
<tr>
<td>Yes, Implemented Jira to track the progress and issues.</td>
<td></td>
</tr>
<tr>
<td>I have not</td>
<td></td>
</tr>
<tr>
<td>Event based communication between servers</td>
<td></td>
</tr>
<tr>
<td>Nope</td>
<td></td>
</tr>
<tr>
<td>A personnel will download the license and pushed to update in local systems</td>
<td></td>
</tr>
<tr>
<td>More show and tell sessions to educate. Daily stand-up meetings</td>
<td></td>
</tr>
</tbody>
</table>

### How do you manage project documentation and knowledge sharing in distributed teams?

23 responses

<table>
<thead>
<tr>
<th>Method</th>
<th>Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>A variety of means</td>
<td>3 (13%)</td>
</tr>
<tr>
<td>Documenting minutes</td>
<td></td>
</tr>
<tr>
<td>PPT and present it</td>
<td></td>
</tr>
<tr>
<td>Share points</td>
<td></td>
</tr>
<tr>
<td>Through wiki eg. Confluence</td>
<td>1 (4%)</td>
</tr>
<tr>
<td>Through integrated tools</td>
<td></td>
</tr>
<tr>
<td>Use tools such as</td>
<td></td>
</tr>
</tbody>
</table>

---

96
What do you think are the emerging trends in distributed software development and distribution?

17 responses

Microfrontends, MicroBackend, DevOps etc

Distributed systems are undergoing a period of significant change and this can be traced back to a number of influential trends: o the emergence of pervasive networking technology; o the emergence of ubiquitous computing coupled with the desire to support user mobility in distributed systems; o the increasing demand for multimedia services; o the view of distributed systems as a utility.

Do not know

Version control and automation

Test driven development, that will help to reduce the bugs.

I am not sure

Micro service with cloud native software support

More frequent version updates, zero downtime upgrades.

Please provide any additional comments, insights, or suggestions related to distributed software development and distribution.

8 responses

Need to make sure all team members are working towards the same goal. Reviews of code play's an important in development, so healthy arguments always help to improve the quality of products.

None that I can think of

Developing using different programming languages poses big challenge during maintenance. Even though it gives flexibility to the developer to build services in the language he likes, the issue arises while maintaining the software especially when a resource leaves the company. I would suggest using minimal and widely used languages like, java, python, etc

Security and Compliance related to be give at most importance and integrated automated solutions to minimise maintenance issues and blockers.

Here are some key point
1. Embrace a Culture of Communication and Collaboration
2. Documentation is Key
3. Security First
4. Distributed Version Control