Software Development
Group A Lyngby
30/10/2024

Development of Large Scale Systems

Assignment 4

Name	Mail
Andreas Fritzbøger	cph-af167@cphbusiness.dk
Owais Dashti	cph-od42@cphbusiness.dk

cphbusiness
COPENHAGEN BUSINESS ACADEMY

Contents

1. Introduction 3
2. SAFe 3
2.1. What is meant by “Essential SAFe” 3
2.2. What is an ART – why is it important? 3
2.3. There are 7 core competencies in SAFe – what are they? 3
2.4. What are 3 benefits of SAFe? 4
2.5. What could be the downsides of SAFe? 4
2.6. What in SAFe is the difference between a user story and an enabler story? 5
2.7. How can XP be used in SAFe teams, what roles does it play and what stages? 5
2.8. SAFe is a Lean Agile framework – what does Lean mean here? 5
2.9. In PI planning risk is measured using ROAM – what does ROAM mean? 6
2.10. How does the Product Owner fit into the ART team? Focusing on the role in relation to the developers – what does the PO do? 6
2.11. What is the SAFe DevOps health assessment and how might it be useful? 7
3. The Spotify Model 7
3.1. Explain why The Spotify Model is much less formal than SAFe 7
3.2. How is it a way of working with fewer events and more focus on team and team coordination? 8
3.3. Both Scrum and Kanban can be used with the Spotify model. Consider using one of them for your exam project, or even a combination of them. What are the main differences between Scrum and Kanban? 8
3.4. What is a Guild and how could you use Guilds in the exam project development process? 9
4. Remote (distributed) Teams 9
4.1. What are the main benefits of working in distributed teams? 9
4.2. What are the main challenges of working in distributed teams? 10
4.3. What practices can help a remote collaboration culture to stay healthy? 11
4.4. What considerations do your group have on remote work in the exam-project period? . 11

1. Introduction

For the fourth assignment in Development of Large Systems, we will be answering questions associated with three topics.

The topics are SAFe, the Spotify Model, and Remote (distributed) Teams.

2. SAFe

2.1. What is meant by "Essential SAFe"

It means that the configuration provides the minimal elements necessary for Agile Release Trains (ARTs) to deliver solutions and is the simplest starting point for implementation. It is the first layer in SAFe.

Essential SAFe is the starting point of SAFe that focuses on the core elements required for SAFe implementation. It provides basic structure for organizing teams around value delivery using ARTs and is designed to support small-to-medium sized organizations or specific teams within larger organizations. It includes two key components:

1. **Agile Teams**: small cross-functional teams that deliver value in short iterations.
2. **Program Level**: focuses on aligning multiple teams, managing dependencies, and coordinating their efforts vi Program Increment (PI) Planning.

Essential SAFe ensures simplified, lightweight approach for organizations looking to adopt agile at scale.

2.2. What is an ART – why is it important?

An **Agile Release Train (ART)** in SAFe is a key organizational structure that consists of long-lived, cross-functional team of teams (usually 50-125 people) working together to deliver value in a synchronized way. ARTs operate in time-boxed iterations, typically lasting 8-12 weeks, called **Program Increments (PIs)**, which are essential for planning, building, and delivering features across multiple agile teams.

ART is important because it ensures that all teams are working toward a shared vision and common objectives, aligning their efforts through regular **PI planning** sessions. This alignment helps manage dependencies and keeps everyone focused on the same goals.

Additionally, ARTs enable a continuous flow of value by coordinating the efforts of multiple Agile teams to deliver incremental features. They support **iterative development** and ensure that teams deliver usable solutions frequently. ARTs also enhance communication between teams, stakeholders, and other involved parties, ensuring everyone is aligned and aware of project progress, risks, and dependencies.

2.3. There are 7 core competencies in SAFe – what are they?

The competencies are:

1. **Team and Technical Agility** - It describes the critical skills and Lean-Agile principles and practices that teams use to create solutions for their customers.
2. **Agile Product Delivery** - Which is used to ensure enterprises creates the correct solutions to the correct customers on time. It needs a balance for execution focus with a customer focus. The competency is a customer-centric approach to define, build, and release a continuous flow of good products and services to the customers and users.

3. **Enterprise Solution Delivery** - It describes using Lean-Agile principles and practices to specify, develop, deploy, operate, and evolve the best software applications, networks and other sophisticated systems in the world.
4. **Lean Portfolio Management** - Which is used to address, why chosen solutions are required. The competency uses strategy and execution by applying Lean and systems thinking.
5. **Organizational Agility** - It helps businesses address how they can quickly respond to challenges and opportunities that gets presented by today's rapidly evolving markets. The competency helps the businesses to become more flexible and adaptable for new challenges and requirements.
6. **Continuous Learning Culture** - Which has a set of values and practices that encourage people and enterprises to continually increase their knowledge, competence, performance, and innovation. The competency helps business be prepared, for when they enter new territory.
7. **Lean-Agile Leadership** - It is used to help the leaders in the organization to internalize and model leaner ways of thinking and operation so that team members can learn from their example, coaching, and encouragement. This will in the end help the teams flourish and produce great values and results.

2.4. What are 3 benefits of SAFe?

The Scaled Agile Framework (SAFe) offers several benefits that help organizations scale Agile practices effectively across large teams and complex systems. The following are 3 key benefits:

1. **Alignment Across Teams and the Organization**: SAFe ensures alignment between multiple teams working toward a common vision by using structures like Agile Release Trains (ARTs) and regular Program Increment (PI) Planning. This alignment helps teams understand their dependencies, goals, and priorities, ensuring that everyone is on the same page. By facilitating cross-team collaboration and communication, SAFe reduces misalignment and helps teams work together more effectively toward shared objectives.

2. **Improved Quality and Faster Delivery**: SAFe promotes frequent releases and incremental value delivery through iterative development. Using techniques like System Demos and continuous feedback loops, teams can spot and fix issues earlier in the process, leading to higher quality products. Moreover, by delivering smaller, usable increments of the product, organizations can respond quickly to changes in customer needs or market conditions, shortening the time to market.

3. **Increased Transparency and Visibility**: One of SAFe's core strengths is the transparency it creates at all levels of the organization. Through tools like Program Boards and PI planning sessions, everyone from teams to executives can track progress, risks, and dependencies in real-time. This visibility leads to more informed decision-making, increased accountability, and a clearer understanding of how each team's work contributes to the larger goals of the organization.

2.5. What could be the downsides of SAFe?

SAFe provides additional layers to oversight, administration and coordination. While the layers make SAFe popular with enterprises, it also make it resemble the waterfall approach, which is not as popular anymore. SAFe does not grant as much freedom compared to scrum, because SAFe demands administrative roles to oversee multiple projects and coordinate releases and dependencies. The developers' freedom resembles that of a waterfall environment, and SAFe can often slow down processes and limit flexibility compared to an agile environment.

SAFe removes developers, testers, and product owners from the decision-making process, which can limit their understanding of the software development lifecycle and hinder their ability to conduct collaborative planning sessions.
SAFe can lead to longer planning periods and more fixed roles within development cycles. These points goes against the idea of delivering in short sprints to get new software faster and creating a continuous loop to ensure quality at each step on the way.

2.6. What in SAFe is the difference between a user story and an enabler story?

In SAFe, User Stories and Enabler Stories serve distinct purposes, although both are types of work that teams complete during development.

- **User Story**: A User Story is a small, functional piece of work that represents a feature or functionality from the end-user's perspective. It describes what the user needs and why, focusing on delivering direct value to the customer. For example, a user story might state, "As a customer, I want to be able to reset my password so that I can regain access to my account." User stories are essential for creating user-facing features and are often linked to business objectives.
- **Enabler Story**: An Enabler Story, on the other hand, does not directly deliver user-facing value but is necessary to support the architecture, infrastructure, or technical work required for future development. Enabler stories focus on things like technical debt, research, compliance, or improving system performance. For instance, "Refactor the authentication module to support multi-factor authentication" would be an enabler story. While it doesn't deliver immediate value to the end user, it enables future features or improvements.

The main difference lies in their focus: User Stories deliver direct value to the customer, while Enabler Stories support the technical or architectural needs to enable smoother, more efficient development in the future. Both are critical to balancing the delivery of features and maintaining a healthy, scalable system.

2.7. How can XP be used in SAFe teams, what roles does it play and what stages?

Extreme Programing (XP) can be used to improve the teams performance and to ensure, they are solving the right problems. It enhances performance, by fostering collaboration, continuous improvement, and feedback-driven development.
It is most optimal to use XP in the early stage of development. Here the teams can establish practices that ensure technical excellence and customer alignment from the start.
XP can also be useful in later iterations by promoting a continuous learning culture and maintaining high-quality standards as the system evolves. Though it is most effective early on, aspects such as continuous refactoring and ongoing customer collaboration can help keep teams on track throughout the development cycle.

2.8. SAFe is a Lean Agile framework – what does Lean mean here?

In SAFe, "Lean" refers to principles derived from Lean thinking and Lean manufacturing, emphasizing efficiency and the continuous flow of value to the customer while minimizing waste. The goal of Lean in SAFe is to maximize value delivery by eliminating any activities that do not directly contribute to customer outcomes.

Lean focuses on maximizing flow by ensuring that work moves quickly and smoothly through the development process without unnecessary delays. This is done by aligning teams around

Value Streams, where the focus is on continuously delivering small, incremental values to customers. Another critical aspect of Lean is the emphasis on waste reduction—anything that doesn't add value is considered waste. This could be inefficiencies like excessive documentation, rework, or waiting times between teams that hinder progress.

Lean also promotes a culture of continuous improvement. Teams regularly assess their processes through retrospectives and make small, iterative changes to enhance productivity and quality. This philosophy, often referred to as Kaizen, encourages ongoing learning and adaptation to improve efficiency over time.

In summary, Lean in SAFe ensures that teams focus on delivering value quickly and efficiently, reducing waste, and fostering a culture of improvement, leading to higher customer satisfaction and better business outcomes.

2.9. In PI planning risk is measured using ROAM – what does ROAM mean?

ROAM is a method, which is an abbreviation for Resolved, Owned, Accepted, and Mitigated, and are categories in which risks are put into for effective management.
The four categories are described as:

- **Resolved** - The risk is not a problem.
- **Owned** - A member of the team takes ownership of the item, as it was not resolved in the meeting.
- **Accepted** - The risk cannot be resolved, so it has to be understood and accepted, for what it is.
- **Mitigated** - Formulate a plan to eradicate the risk.

2.10. How does the Product Owner fit into the ART team? Focusing on the role in relation to the developers – what does the PO do?

In the context of an Agile Release Train (ART) within SAFe, the Product Owner (PO) plays a crucial role that focuses on bridging the gap between stakeholders and the development team. The PO is responsible for maximizing the value of the product resulting from the work of the ART team and plays an integral part in ensuring that the developers have a clear understanding of the product vision, priorities, and user needs.

A key aspect of the PO's role is managing the Team Backlog. This involves defining and prioritizing user stories and enabler stories to ensure that developers are always focused on delivering the most valuable features. The PO organizes the backlog, ensuring that the stories are well-organized, with items clearly articulated and aligned with the overall goals of the ART. This prioritization allows developers to work efficiently, knowing that they are contributing to the most critical objectives first.

The PO collaborates closely with developers throughout the development process. This collaboration includes clarifying requirements, answering questions, and providing feedback on features. By maintaining open lines of communication, the PO ensures that developers fully understand the needs of end users and can implement features that address these needs effectively. The PO also participates in Program Increment (PI) Planning sessions, articulating the product vision and upcoming objectives to the team. This helps maintain alignment between the developers' day-to-day work and the broader goals of the ART.

An essential function of the Product Owner is to define acceptance criteria for user stories. Once a feature is developed, the PO reviews it to ensure that it meets the predefined criteria and delivers the expected value. Only when the work is validated by the PO, is it considered complete and ready for release. This continuous review process helps maintain the quality and relevance of the product.

Furthermore, the PO acts as an intermediary between the development team and various stakeholders such as customers, business owners, and other teams. The PO gathers feedback from stakeholders and integrates it into the backlog, ensuring that the product remains aligned with user needs and business goals. Any changes in priorities or direction are communicated clearly to the developers, ensuring that the team remains focused on what matters most.

2.11. What is the SAFe DevOps health assessment and how might it be useful?

SAFe DevOps has a tool called **Health Radar**, which is a comprehensive assessment tool that helps organizations evaluate their DevOps capabilities over four dimensions. The dimensions are

- Continuous Exploration
- Continuous Integration
- Continuous Deployment, and
- Release on Demand.

Each dimension has four sub-dimensions.

To use Health Radar, the teams first conduct honest assessment of their current practices in each area. This is done in a workshop setting, which involves representatives from all relevant roles. The team reviews the defined practices and determines, which level best represents their current state.

The results are plotted, creating a visual of the DevOps maturity, whereafter the results are analyzed. With the analyzation, the team can assess and identify areas for growth and track progress over time. Health Radar helps teams and organizations navigate the journey of DevOps transformation, and help teams continuously improve their ability to deliver value to customers.

3. The Spotify Model

3.1. Explain why The Spotify Model is much less formal than SAFe

The Spotify Model is much less formal than SAFe because it focuses on autonomy, flexibility, and team-driven processes rather than adhering to a strict framework or structure. While SAFe provides a comprehensive, well-defined structure for scaling agile practices across large organizations, with roles, ceremonies, and governance explicitly outlined, the Spotify Model emphasizes a more decentralized and adaptable approach.

One key difference is that the Spotify Model allows teams, known as Squads, to have greater autonomy in how they operate. Each Squad in the Spotify Model functions as a mini-startup with the freedom to decide its tools, processes, and goals. While there are overarching objectives and guidance from Tribes (groups of Squads working towards a common mission), the day-to-day working methods are largely left up to the teams themselves. This autonomy contrasts with SAFe, where the roles and responsibilities are much more defined, and alignment across teams is achieved through formal structures like Agile Release Trains (ARTs) and Program Increment (PI) Planning.

Another reason for the informality of the Spotify Model is its focus on informal communication and lightweight governance. In Spotify, alignment is often achieved through informal syncs, cross-team collaboration, and self-organized communities of practice known as Guilds and Chapters, where knowledge-sharing is encouraged. SAFe, on the other hand, prescribes regular ceremonies like PI Planning, Inspect & Adapt workshops, and daily stand-ups to ensure coordination and alignment.

Additionally, the Spotify Model encourages experimentation and iterative learning at a faster pace, without the constraints of formal processes. Teams are encouraged to try new things, fail fast, and adjust without the need for heavy documentation or formal approval processes. SAFe, while agile at scale, requires a more structured approach with clear definitions of value streams, roles like Product Owners, Scrum Masters, and RTEs (Release Train Engineers), and a consistent cadence of planning and execution.

In essence, the Spotify Model thrives on flexibility, team autonomy, and fluid communication, making it much less formal compared to the structured, scalable, and coordinated approach of SAFe. This informality gives Spotify teams the freedom to innovate rapidly but may also lead to challenges in maintaining consistency and alignment across larger organizations, something that SAFe addresses with its formalized roles and processes.

3.2. How is it a way of working with fewer events and more focus on team and team coordination?

Spotify focuses on organizing teams into groups, which are part of or consists of other groups. The biggest focus is team autonomy and Spotify relies on people to reach objectives and achieve goals.

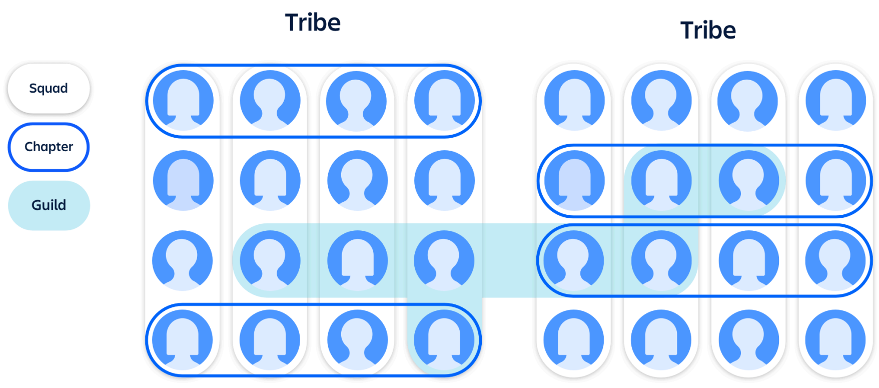

Figure 1: Overview of the organization structure, with groups within groups, and people who can belong to multiple groups.

3.3. Both Scrum and Kanban can be used with the Spotify model. Consider using one of them for your exam project, or even a combination of them. What are the main differences between Scrum and Kanban?

Scrum and Kanban are both popular agile methodologies, but they differ in, how they manage work and structure teams.

Scrum is a structured framework that operates in fixed-length iterations called sprints, typically lasting 2 to 4 weeks. During each sprint, the team commits to completing a specific set of tasks, known as the sprint backlog. Scrum defines clear roles—Product Owner, Scrum Master, and Development Team—and includes specific ceremonies like Sprint Planning, Daily Stand-ups, and Retrospectives. This approach offers a high level of organization, with teams delivering a potentially shippable product increment at the end of each sprint. Progress is measured by velocity, which tracks how much work the team completes during each sprint.

Kanban, on the other hand, is more flexible and does not rely on time-boxed iterations. Instead, it emphasizes a continuous flow of work, allowing tasks to be pulled through the workflow as capacity becomes available. Kanban doesn't require specific roles or ceremonies, making it less formal and easier to integrate into existing workflows. It focuses on limiting Work in Progress (WIP) to ensure teams don't take on more tasks than they can handle at once. Progress is tracked through metrics like cycle time and lead time, which measure how long it takes for a task to move through the workflow.

The key difference is that Scrum offers a structured, time-bound approach with predefined roles and meetings, while Kanban provides more flexibility, allowing work to flow continuously with fewer formalities. Teams can choose between the two depending on whether they need more structure and regular sprint cycles (Scrum) or a more adaptable and fluid process (Kanban).

Scrum and Kanban can be combined, and this approach is often referred to as Scrumban. By merging these methodologies, teams can benefit from the structured, time-bound nature of Scrum while also incorporating the flexibility and flow of Kanban.

In a Scrumban approach, teams still work in sprints and may have regular planning, review, and retrospective meetings, as in Scrum. However, they adopt Kanban's Work in Progress (WIP) limits and visual task boards, which help improve workflow management and prevent teams from being overwhelmed by too many tasks at once. Instead of strictly adhering to Scrum's sprint backlog, Scrumban allows tasks to be pulled into the process as needed, providing greater adaptability while still maintaining some of Scrum's structure.

This combination allows teams to adjust based on their evolving needs, making Scrumban a practical choice for teams that want both the predictability of Scrum and the flexibility of Kanban.

3.4. What is a Guild and how could you use Guilds in the exam project development process?

A guild is a group of people, who are specialist in specific topics. Anyone in the organization can be part of a guild, no matter what tribe they are from. A guild does not have a leader, but instead a Guild Coordinator, who helps bring people together.
For our exam project, we can use Guilds to help other groups in the exam project development process. If none our the team/tribe are well known about the topic, the we ourselves can look for help from someone in the Guild.

4. Remote (distributed) Teams

4.1. What are the main benefits of working in distributed teams?

Remote software development teams, or distributed teams, have become a widespread strategy in modern organizations, especially following the COVID. While this offers significant benefits, it also has challenges that organizations must address to ensure smooth collaboration and high productivity. The conversation surrounding distributed teams gained further attention when prominent figures like Elon Musk and Sam Altman described remote work as an "unsuccessful experiment.” Yet, many companies have found that distributed teams can thrive under the right conditions.

One of the key advantages of distributed teams is access to a global talent pool. Companies no longer need to limit their hiring to a specific geographic area, opening up opportunities to recruit skilled developers from international tech hubs like India, Ukraine, and Brazil. This not only increases diversity but also improves the quality of hires by allowing businesses to select the best talent, regardless of location.

Additionally, remote teams often experience increased focus and fewer distractions compared to in-office teams. A study by Harvard Business Review found that remote workers made 13.5% more calls than their office-based counterparts, largely due to fewer interruptions like commuting, coffee breaks, and casual office conversations. This increased productivity, combined with cost savings from reduced office overheads, makes remote development teams an attractive option for many companies.

Another significant benefit is higher employee satisfaction and flexibility. Remote work allows team members to create a work-life balance that suits their personal needs, such as traveling, working during different hours, or spending more time with family, without compromising their job performance. This flexibility can lead to greater employee retention and agility, as long as deadlines are met and the work gets done.

4.2. What are the main challenges of working in distributed teams?

The challenges of working in distributed teams are:

- Coordinating the work across time zones.
- Building rapports when everyone is not in the same office.
- Collaborating among different cultures.
- Scheduling meeting or informal conversations, when both teams are online at the same for only a few hours.

It can be difficult to cooperate and coordinate on a project as a team, when all team members are not physically present.

To overcome these challenges, organizations need to establish clear communication channels and documentation practices. Relying solely on video meetings can lead to burnout and inefficiency, so it’s essential to mix synchronous communication (e.g., meetings) with asynchronous methods (e.g., detailed documentation and project updates). Creating a centralized knowledge base or wiki ensures that all team members have access to important information and can contribute to shared resources.

Building a strong company culture in a remote environment is another ongoing challenge. Maintaining a sense of collaboration and connection between team members requires deliberate effort, such as organizing virtual meetings with cameras on, creating non-work-related chat channels, and providing honest feedback in a constructive manner. In some cases, arranging in-person meetings or visits to remote offices can strengthen team bonds and improve collaboration.

Trust is a critical component of remote team management. Leaders must trust their employees to complete their work without constant oversight, while employees need to feel that their needs are being heard and addressed. Without trust, remote work strategies may fail due to micromanagement or misaligned expectations.

4.3. What practices can help a remote collaboration culture to stay healthy?

- **Clear Communication**: Establishing a transparent and regular communication channels using tools like Discord, Slack, Zoom, or Microsoft Teams. This ensures that everyone is on the same page, reducing misunderstandings.
- **Regular Check-ins**: Scheduling routine team meetings and one-on-ones to foster connection and address any issues early. These can be daily stand-ups or weekly check-ins, promoting a sense of belonging and accountability.
- **Setting Expectations and Goals**: Defining clear goals, deadlines, and responsibilities. This helps remote teams stay aligned and ensures that everyone knows what is expected of them.
- **Social Interaction**: Creating opportunities for casual, non-work-related interactions through virtual coffee breaks, team-building activities, or informal chat channels to maintain personal connections and team morale.
- **Providing Flexibility**: Allowing team members to manage their work schedules with flexibility, as remote work offers different time zones and personal circumstances. This demonstrates trust and respect for individual work-life balance.
- **Using Collaborative Tools**: Implementing collaborative platforms like Google Workspace, Notion, or Trello to organize work, track progress, and share resources efficiently.
- **Feedback**: Promoting an open feedback culture where employees feel comfortable sharing their thoughts and suggestions. This helps resolve issues quickly and encourages continuous improvement.
- **Training and Development**: Providing training in remote work tools, team collaboration techniques, and soft skills to help employees thrive in a remote environment.
- **Recognizing Achievements**: Celebrating team successes, individual accomplishments, and milestones to keep morale high and create a positive work environment.

4.4. What considerations do your group have on remote work in the exam-project period?

We are a small team consisting of two members. This means there are less time schedules to coordinate with. However, we are dependent on all team members to be present, when we need to do planning and group-work

Our group will primarily work remotely, since we are students and do not have access to an office. The one place we can meet at is the school, but here we are dependent on free rooms, which other students definitely also wants to use for quiet workspace.

To conclude, we will primarily work remotely and communicate via Discord. When we feel it's necessary for us to meet in person, we will do it at the school and have to try our luck for free a room.