Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Optimizing Software Development Through Flow Metrics Analysis in the Scaled Agile Framework (SAFe)

Achmad Fathurrazi Akbar[1)], Eko Indrajit[2)], Amelia Makmur[3)], Handri Santoso[4)]
[1)2)3)4)]Pradita University, Serpong, Tangerang
[1)]Achmad.fathurrazi@student.pradita.ac.id, [2)]eko.indrajit@pradita.ac.id, [3)]amelia.makmur@pradita.ac,id,
[4)]handri.santoso@pradita.ac.id

Submitted : Feb 5, 2025 | **Accepted** : March 23, 2025 | **Published** : April 15, 2025

Abstract: The market development is increasingly dynamic; it is very important to build an effective strategy given the increasing complexity of software development. With the support of project management methodologies, the integration of Agile concepts is a practical approach that demands that it is always adaptive to respond to change, especially in managing large-scale projects involving many cross-functional teams and departments. The application of the Scaled Agile Framework (SAFe) is one of the practices that support organizations in managing projects and product development. This research focuses on measuring productivity, efficiency, and alignment in software development with Flow Metric techniques, which include Flow Velocity, Flow Efficiency, Flow Time, and Flow Load. This research uses qualitative methods, combining literature analysis, semi-structured interviews with Agile practitioners, and observation of the case by comparing two interdependent Program Increments (PIs). This research has shown significant improvements after implementing Flow Metric into SAFe, with a 106%25% increase in Flow Velocity, 49.33% increase in Flow Efficiency, 44.81% decrease in Flow Time, and 39.47% reduction in Flow Load. The results also highlight the importance of other support such as incorporating Continuous Improvement and Continuous Development (CI/CD) pipelines, backlog synchronization, and PI Planning to optimize processes and add value to the organization. The results of this study provide practical insights and serve as an important reference for organizations looking to implement the SAFe framework and the Flow Metrics technique techniques to efficiently navigate complexity.

Keywords: Agile Project Management, SAFe, Flow Metrics, Continuous Delivery, Software Development Efficiency

INTRODUCTION

In this modern digital delivery environment, complex software development requires adaptive approaches and strategies to respond to the market. With a wide range of best practices, Agile is an approach that emphasizes iteration, flexibility and collaboration that brings benefits in product development and project management focused on value, customer satisfaction, and a dynamic environment.

Despite these benefits, Agile implementations in large-scale organizations often face risks and constraints, such as inconsistent development processes, poor priority management, fragmented workflows between teams, excessive work-in-progress (WIP), and no visibility of team activities. These constraints resulted in delivery delays, decreased team morale, and low value of outputs produced which led to decreased customer satisfaction and business performance.

Addressing this issue, the Scaled Agile Framework (SAFe) is one of the solutions adopted for large and complex projects involving various cross-functional and cross-departmental teams. By integrating Agile principles into structured organizational processes, SAFe enables teams to be more adaptive to changing needs and improve value delivery (Chahal, 2023). However, SAFe implementation also presents its own challenges, such as a lack of visibility of project progress, uneven workload distribution, and difficulty in detecting bottlenecks early on. This often leads to uncertainty among managers and teams, which results in delays of important features, increased stress, and general dissatisfaction.

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Therefore, this study addresses a critical issue: the lack of visibility and effective measurement in the current implementation of SAFe, which hinders timely decision making, workload management, and overall project effectiveness. To address this problem, the use of Flow Metrics—specifically flow velocity, flow efficiency, flow time, and flow load—is proposed as a targeted solution. Flow Metrics provides clear and measurable insights to teams and management about their performance, enabling the proactive identification and resolution of bottlenecks, optimization of task distribution, and increased strategic alignment between operational tasks and business objectives.

The importance of this research lies in the attempt to bridge the gap between theoretical knowledge and the practical reality of SAFe implementation. Although the benefits of SAFe have been well documented conceptually, its real-world applicability remains a challenge. By empirically exploring and validating the effectiveness of Flow Metrics in SAFe, this research contributes valuable and actionable insights. Practically, these insights will empower project managers and Agile practitioners to effectively overcome common implementation obstacles, increase productivity, improve project outcomes, and drive greater team satisfaction. Strategically, the findings of this study will guide organizations in aligning their Agile practices with business objectives, ensuring consistent and high-quality delivery of value in response to dynamic market demands.

Thus, this research makes both academic and practical contributions, directly benefitting organizations striving for success amid technological change and increasingly fierce market competition.

LITERATURE REVIEW

Project Management: Agile Methodology

Agile methodology has grown in importance as a business and software development technique, mostly because of its capacity to react swiftly to shifting project dynamics and needs. Agile emphasizes iteration, teamwork, and quick user input to enable more customer-focused and responsive product development (Nuottila et al., 2022).

A group of working software developers released the Agile Manifesto in 2001, which is one of the pillars of the Agile approach. The Manifesto highlights four core values: Individuals and interactions over processes and tools; Working software over comprehensive documentation; Customer collaboration over contract negotiation; Responding to change over following a plan (*Agile Alliance*, n.d.).

Twelve guiding principles are also included in the Agile Manifesto, which emphasizes the value of regular software delivery, close stakeholder and developer involvement, and prioritizing change as a component of the development process (*Agile Alliance*, n.d.). These guidelines give teams a structure for more adaptable and responsive work, which is crucial for software projects since these projects frequently encounter new difficulties and shifting requirements.

Comprehensive guidelines for effective project management procedures are provided by the Project Management Body of Knowledge (PMBOK). Agile approaches and the PMBOK's suggested practices and principles can be combined to produce a more useful project management strategy. Project management requires an adaptable and goal-oriented methodology. For the project to be able to adjust to changing circumstances, the following project management principles should be used: transparent communication, change management, and ongoing planning (Project Management Institute, 2021).

Various frameworks within the Agile methodology have been designed to meet business needs and the unique nature of projects. Extreme Programming (XP), Scrum, Kanban, Lean, Scaled Agile Framework (SAFe), and others are among the frameworks often applied. Software development is organized in iterative cycles called sprints by the Scrum framework, for example, where teams strive to complete a predetermined set of tasks within a given time. To increase productivity, Kanban, on the other hand, emphasizes continuous task management and workflow visualization (Pinciroli, 2024). By combining Agile concepts with frameworks, SAFe can be applied by organizations to oversee projects consisting of various cross-functional and cross-departmental teams, thereby increasing the organization's capacity to adapt to change and increase value (Cvejič, 2022).

Scaled Agile Framework (SAFe)

In order to improve collaboration, efficiency, and value delivery to clients, Scaled Agile Framework (SAFe) was developed to help enterprises manage projects involving multiple departments and teams (Gustavsson et al., 2022). SAFe offers a clear framework for project planning, execution, and control at the corporate level by combining several Agile and Lean methodologies.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

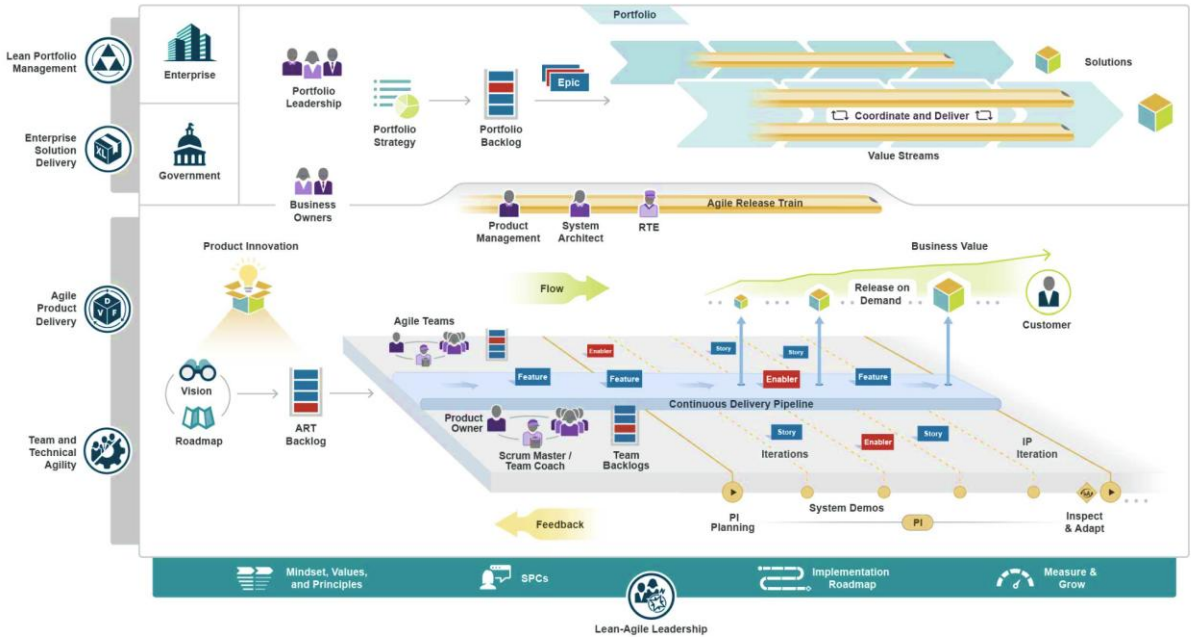

Fig. 1 Overview of SAFe Framework (*Scaled Agile Framework*, n.d.)

Figure 1 illustrates the entire Scaled Agile Framework (SAFe) includes a number of key components, from Lean Portfolio Management, which is responsible for strategic direction and investment management in the form of Epics, to Enterprise Solution Delivery, which manages cross-departmental coordination and Value Streams to produce high-value solutions. The Agile Product Delivery component includes the Agile Release Train (ART), which is a combination of Agile teams that work collaboratively in short iterations or sprints, supported by important roles such as Product Owner, Scrum Master, and Release Train Engineer (RTE), as well as the use of a Continuous Delivery Pipeline for automatic integration and deployment. Meanwhile, Team and Technical Agility ensures smooth collaboration and transparency through the Kanban method, the use of well-managed backlogs, and iterative product feature development.

In addition, SAFe also emphasizes Product Innovation through a clear product vision and roadmap, Program Increment (PI) Planning for periodic coordination, and Inspect & Adapt for continuous evaluation. The Continuous Delivery Pipeline component includes Continuous Integration (CI) and Continuous Deployment (CD) so that products can be released at any time according to customer needs (Release on Demand). This framework is also supported by Lean-Agile Leadership, which builds Agile mindsets, values, and principles in the organization, as well as SAFe Program Consultants (SPCs) who help with effective implementation. Finally, SAFe is complemented by an Implementation Roadmap as a step-by-step guide to implementation and Measure & Grow, which utilizes performance metrics to ensure continuous improvement based on direct feedback from customers.

A detailed explanation of each SAFe component is provided in Table 1 below:

Table 1.
Explanation of Scaled Agile Framework (SAFe) Component

Main Component	Sub-Component / Activity	Description
Lean Portfolio Management	Enterprise	Refers to the overall organizational structure that implements Agile strategies to achieve business objectives.
	Portfolio Leadership	Team or individual responsible for strategic decision making, investment management, and ensuring alignment between strategy and execution.
	Portfolio Strategy	Explains clear strategic direction and priorities, forming the basis for activities at the operational level.
	Portfolio Backlog	List of strategic initiative priorities (Epic) that must be carried out to achieve organizational targets.

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Main Component	Sub-Component / Activity	Description
	Epic	A large-scale strategic initiative or project that provides significant benefits towards achieving organizational goals.
Enterprise Solution Delivery	Solution	Products or services produced by organizations, especially those that are large-scale or highly complex.
	Coordinate and Deliver	Cross-departmental coordination activities to ensure effective integration of solutions and timely delivery.
	Value Streams	The set of interdependent activities, from planning to delivery of the final product, that generate direct value for customers.
Government	Regulatory / External Entities	Involvement of external parties, including governmental regulations or industry standards that affect the process and results of product development within the organization.
Agile Product Delivery	Agile Release Train (ART)	A combined team of several Agile teams that work collaboratively and integrated to achieve common goals on a large scale.
	Product Management	The role that sets the priority of product features, ensuring alignment with market and customer needs.
	System Architect	Responsible for the technical design and architecture of the product to ensure the integration and overall quality of the solution.
	Product Owner (PO)	Responsible for managing the product backlog, ensuring feature development according to customer needs.
	Scrum Master/Team Coach	Team facilitator who ensures the Agile process runs smoothly, collaboratively, and efficiently.
	Release Train Engineer (RTE)	Agile Release Train Leader, manages work integration between teams and ensures the effectiveness of the Program Increment (PI).
	ART Backlog	List of priority product features to be developed by the team in the Agile Release Train.
	Continuous Delivery Pipeline	Automated infrastructure for continuous integration (Continuous Integration/CI), continuous deployment (Continuous Deployment/CD), and release on customer demand.
	Iterations (Sprint)	Short work cycle (usually 2–4 weeks) in which the Agile team completes the planned feature.
	System Demos	Regular presentations of Agile team work results to validate the progress of features developed each iteration.
	PI Planning	Periodic planning with the entire Agile team to determine the work to be completed in one Program Increment (8–12 weeks).
	Inspect and Adapt	Final evaluation activity of the Program Increment to identify continuous improvements to processes and products.
Team and Technical Agility	Agile Teams	A small team of cross-functional individuals who work collaboratively to iteratively produce products.
	Kanban	A workflow visualization method for monitoring task progress and limiting the

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

Main Component	Sub-Component / Activity	Description
		amount of Work-In-Progress (WIP) to prevent bottlenecks.
	Flow	The concept of a smooth flow of work value without obstacles in the product development process.
	Team Backlogs	A list of tasks that the team must complete in a given iteration.
	Features	Product functions that have direct value for end users.
	Enablers	Technical or preparatory tasks that support the development of future product features.
	Stories	A simple description that contains the tasks or user requirements that will be developed in a particular iteration.
Product Innovation	Vision	A broad overview of the long-term goals and direction of the product.
	Roadmap	Detailed product planning, showing priorities and schedules for feature development in the medium to long term.
	ART Backlog	Priority of tasks and features prepared for development by the Agile Release Train to achieve maximum product innovation.
Continuous Delivery Pipeline	Continuous Integration (CI)	The process of merging code or features that are developed continuously, automatically, and quickly.
	Continuous Deployment (CD)	Processes automate the deployment of products into the production environment quickly and safely.
	Release on Demand	The organization's ability to release features at any time according to market or customer needs without having to wait for regular release cycles.
Program Increment	PI Planning	A massive planning event involving the entire team in the Agile Release Train to determine work for a specific period (8-12 weeks).
	IP Iteration (Innovation & Planning Iteration)	A special time at the end of the Program Increment to evaluate, innovate, experiment, and plan the next iteration.
	System Demos	Regular work demos to evaluate product progress.
	Inspect & Adapt	Evaluate the entire Program Increment (PI) cycle to identify challenges and opportunities for further improvement.
Feedback	Feedback from Customers	Information or customer responses to developed features form the basis for continuous product improvement.
Lean-Agile Leadership	Mindset	Agile and Lean perspectives adopted by the organization in work and decision making.
	Values	Core values such as transparency, collaboration, and continuous improvement.
	Principles	Lean-Agile guiding principles that underlie all activities in SAFe.
	SPCs (SAFe Program Consultants)	Professionals who are experts in SAFe implementation, responsible for supporting organizations in the effective adoption of Agile.
Measure and Grow	Metrics and Continuous Improvement	Use a variety of metrics (such as Flow Metrics) to monitor performance, identify areas for

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Main Component	Sub-Component / Activity	Description
		improvement, and adjust development processes on a regular basis.
Implementation Roadmap	Roadmap for adopting SAFe	A step-by-step guide for organizations to systematically implement SAFe, from preparation to successful execution.

SAFe's ability to handle issues such as team collaboration, dependency management, and risk management that arise in large-scale software development is one of its main advantages. Organizations can manage projects more organization and systematically by using SAFe's several layers, which include Team, Program, Large Solution, and Portfolio (Pandey et al., 2023).

Several industries, including finance, manufacturing, and information technology use SAFe in their product development implementation. Although there are limitations to team autonomy that can negatively impact performance and collaboration, this research shows that implementing SAFe can improve team performance and customer satisfaction (Gustavsson et al., 2022). Therefore, implementing SAFe is important for organizations to understand in the context of their specific needs and requirements.

Flow Metrics

An essential technique for software development using Agile methods, including the SAFe framework, is Flow Metrics. In addition to offering insights into how value is produced and provided to clients, these metrics are used to assess the efficacy and efficiency of the development process. Teams can track project progress, find process bottlenecks, and apply ongoing changes by utilizing flow metrics (Edison et al., 2022).

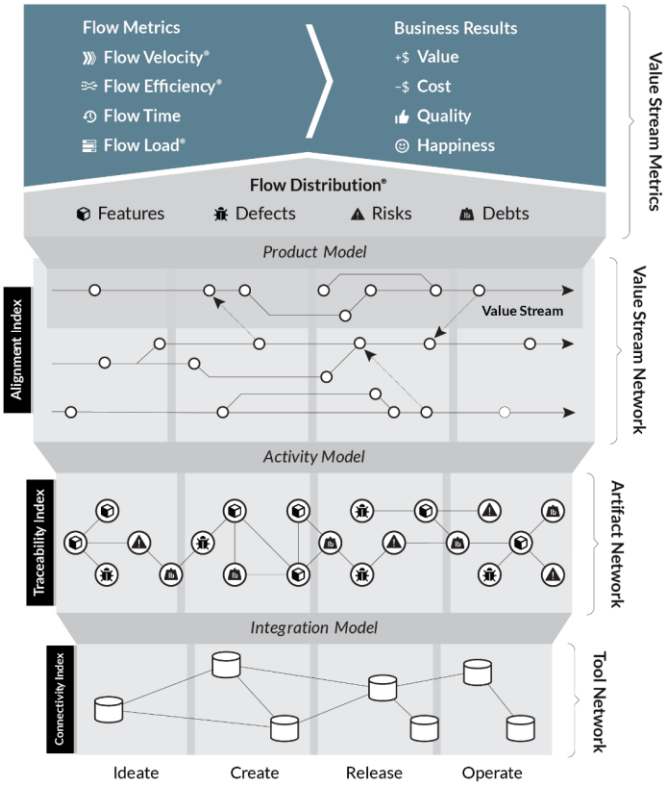

Fig. 2 Flow Metrics (*Flow Framework*, n.d.)

Four indications that can assist businesses in measuring and streamlining their processes are depicted in the Flow Metrics diagram in Figure 2. These include:

1. Flow Velocity gives a summary of the team's productivity by counting the number of tasks such as features, defects, risks, or technical debt finished in a given amount of time.
2. Flow Efficiency finds roadblocks or delays that may impede progress and, measures the ratio of active work time to total work time.

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

3. Flow Time gives a clear picture of process efficiency by calculating the amount of time required to accomplish a task from beginning to end.
4. Flow Load keeping track of the quantity of active jobs, enables the team to efficiently manage capacity and steer clear of an excessive burden.

The four steps mentioned above provide a thorough understanding of process dynamics and facilitate continuous improvement of value delivery. Organizations can prioritize work according to strategic objectives using Flow Velocity, which also offers an overview of the distribution of work by categories including features, defects, risks, and technical debt. Organizations can implement Flow Metrics to link these indications to business objectives such as higher value, lower costs, better quality, and more satisfied customers.

The Flow Framework highlights the importance of Artifact Networks, which describe the relationships of various process elements, and Value Stream Metrics, which assist companies in understanding the overall value network.

Previous Research

Studies on the use of Agile methodology have been carried out across a range of industries, demonstrating the variety of challenges and advantages that businesses encounter while using Agile techniques.

The application of Agile project management in the construction sector has been shown to have several advantages, including improved efficiency and collaboration. However, corporate culture and project complexity present challenges. When applying the Agile approach to building projects, this study highlights how crucial it is to consider all aspects of social, economic, and environmental sustainability (Pinto et al., 2023).

Investigating a Malaysian multinational company's transformation path in the telecommunications industry through the extensive use of Agile methodology. This study emphasizes the challenges and complexities encountered during the adoption process, in addition to the significance of maintaining the sustainability of Agile practices in the face of significant transformations in the digital age. This study provides significant insight into how businesses might overcome challenges while implementing Agile using a qualitative methodology (Bakar & Dorasamy, 2023).

Agile approaches are being used to solve typical problems in the implementation of ERP in the industry. According to research by Wijaya et al., 2019, the use of the Agile approach can improve the functionality of ERP systems and assist firms in managing change more effectively.

Additional studies highlight the significance of Agile methodology in product management, offering actionable suggestions for companies wishing to adopt Agile. This study highlights the difficulties in achieving balance in hybrid systems and provides recommendations for fostering an innovative and cooperative culture in businesses (Chahal, 2023).

Adopting Agile techniques during the design phase of building projects can improve productivity and collaboration in the industry, according to studies. According to Chathuranga et al. (2023), this study demonstrates that architectural firms can apply Agile project management techniques in design, akin to software development, by employing BIM software solutions.

In order to meet the demands of the EPC (Engineering, Procurement, and Construction) sector, the research will address the necessity of Agile methodologies in construction project design management and offer Agile philosophies and practices. According to Abou-Aly et al., 2024, this study emphasizes how crucial it is to modify Agile in the design environment to improve project efficacy and efficiency.

Scrum dan Kanban adalah dua pendekatan Agile yang dapat meningkatkan kepuasan pelanggan dan inovasi produk dalam konteks pengembangan produk. Kesulitan dalam mengadopsi kerangka kerja Agile juga diidentifikasi dalam penelitian ini, bersama dengan strategi yang dapat digunakan perusahaan untuk tetap kompetitif di pasar yang dinamis (Amajuoyi et al., 2024).

METHOD

Research Design

This study uses a qualitative approach with the main objective of exploring and optimizing the software development process through an in-depth analysis of the application of Flow Metrics within the Scaled Agile Framework (SAFe). The qualitative approach was chosen to enable an in-depth understanding of the phenomena that occur during the implementation of SAFe, especially in identifying factors that influence the performance of the development team.

The research design was systematically designed in five main phase, namely:

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

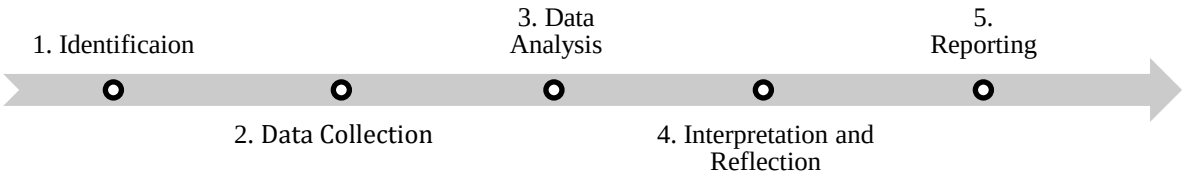

Fig. 3 Research Method

The above phase are structured and logical so that the research results are relevant and related to the main objective, namely optimizing software development using Flow Metrics analysis.

Data Collection

In the data collection phase, three methods were used as follows:

1. Literature Study: The researcher conducted an in-depth theoretical exploration of relevant literature, including scientific publications, reference books, journals, and articles discussing the concepts of SAFe, Flow Metrics, Agile, and best practices in software development. The literature study was used as a theoretical basis and empirical comparison for later field findings.
2. Observation: Observation was carried out on the development team that actively applied SAFe in two interdependent *Program Increments (PI)*. Observation aims to capture the team's actual practices in the field, the constraints faced, and the dynamics that occur during the development cycle. The information from this observation was used to provide empirical context for the data from the interviews.
3. Semi-Structured Interviews: This study used a semi-structured interview method with a total of 16 respondents, consisting of 2 Scrum Masters/Agile Coaches, 2 Product Owners, and 12 Developers who were actively involved in the implementation of SAFe. The reason for using semi-structured interviews is related to the need to extract information and provide freedom for informants to convey their views and experiences. The interviews used a list of questions as an initial guide, but allowed the conversation to develop more broadly, allowing the researcher to gain new insights, explore ideas, and spontaneously identify important issues from the informants.

During the data collection process, ethical considerations were important to ensure consent by clearly explaining the purpose of the study, benefits, and potential risks to the respondents, who volunteered to participate. Researchers protected the confidentiality of respondents by anonymizing the data and using it for academic purposes only.

Data Analysis

The data collected through interviews, observations, and literature studies were qualitatively analyzed using a thematic analysis approach. The analysis process includes the following stages:

1. Data Transcription: Interview recordings are transcribed into text form, supplemented by systematically organized notes on the results of observations.
2. Theme Identification: Codes with similar patterns are grouped into categories, which are then used to find core themes relevant to Flow Metrics and SAFe implementation.
3. Comparative Analysis: The researcher compared data from two different Increment Programs (IP) to evaluate the effect of SAFe implementation on flow metrics (Flow Velocity, Flow Efficiency, Flow Time, Flow Load) and its impact on team productivity.

Interpretation and Reflection

After the data was analyzed, the results were interpreted to gain a further understanding of the implications of the findings for the research objective, namely the optimization of software development using Flow Metrics in SAFe. Data interpretation was carried out by reflecting the findings against relevant theory and practical context in the field.

The researcher explored how significant each flow metric change was after the implementation of Flow Metrics in SAFe, by understanding what the implementation challenges were, and identifying factors that influenced the successful implementation of this technique in real cases. This reflection process is important to ensure that the research results are not only descriptive, but also able to explain the phenomenon analytically and provide practical insights into more efficient software development using SAFe.

Reporting

The final phase in this research is the preparation of a research report that contains a summary of the research process that has been carried out, the results of the analysis found, as well as interpretations and reflections on the

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

data. This reporting section also presents practical implications relevant to Agile practitioners and software project managers, as well as strategic recommendations for organizations that want to implement SAFe effectively using Flow Metrics as a measure of success.

The preparation of research reports aims to ensure that the research process can be clearly followed by readers, making a real contribution to the development of SAFe-based project management practices, as well as providing an applicable guide for companies in optimizing the software development process in a sustainable manner.

RESULT

This research analyzes the impact of applying the Flow Metrics in the Scaled Agile Framework (SAFe), specifically Flow Velocity, Flow Efficiency, Flow Time, and Flow Load. The following presents the results of comparing these metrics before and after the implementation of SAFe, complete with table and graph visualizations to clarify the changes that have occurred.

Flow Velocity

Flow Velocity measures the number of work items completed in a given period of time (for example, one sprint or one Program Increment/PI. These work items include features, bugs fixed, risks addressed, and technical debt resolved. Table 1 shows the team's Flow Velocity data for two sprints before and after the implementation of SAFe.

Table 2.
Flow Velocity Before and After Implementation (per Sprint)

Sprint	Features (Completed)	Defects (Resolved)	Risks (Mitigated)	Debts (Addressed)	Total Items Completed
Program Increment 1 (Before Implementation)					
Sprint 1	4	2	1	1	8
Sprint 2	5	1	1	1	8
Average	4.5	1.5	1	1	8
Program Increment 2 (After Implementation)					
Sprint 1	8	4	2	3	17
Sprint 2	7	5	2	2	16
Average	7.5	4.5	2	2.5	16.5

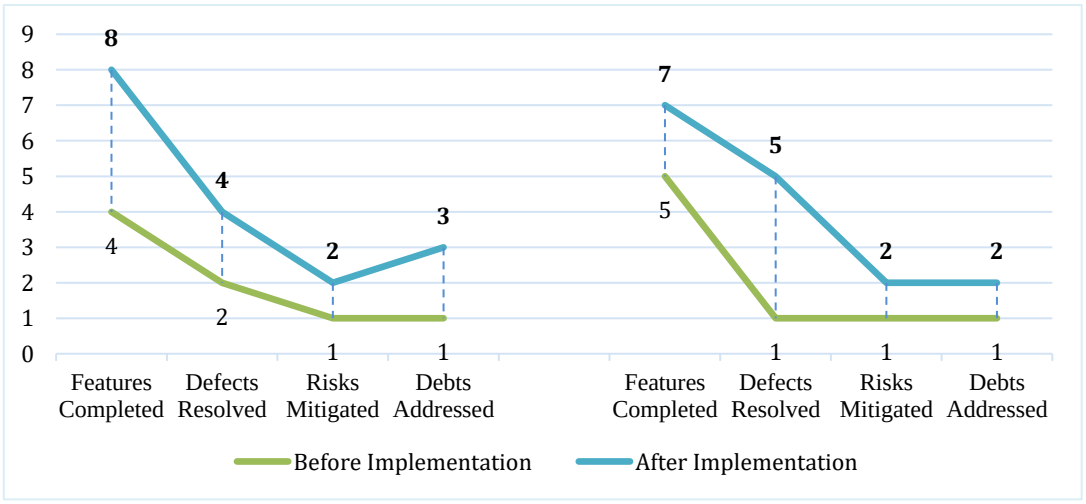

Fig. 3 Flow Velocity Trends Before and After Implementation

In Table 1, it is explained that before implementing SAFe, the team completed an average of 8 work items per sprint. This achievement was dominated by features (around 4.5 items per sprint), while the completion of defects, risks, and technical debt was relatively low (around 1–1.5 items per sprint each). This shows that the team's focus before SAFe was more inclined towards completing key features, while handling defects, mitigating risks, and reducing technical debt tended to be delayed or deprioritized.

Figure 3. Comparison of average Flow Velocity per sprint before and after SAFe implementation. It can be seen that the number of items completed per sprint increased from around 8 to around 16.5 items after SAFe implementation. SAFe implementation resulted in more than double the increase in team productivity (Flow

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Velocity). In addition, the distribution of work completion became more balanced between feature completion, defect resolution, risk mitigation, and technical debt resolution. In other words, after implementing SAFe, the team was not only able to complete more features (~8.5 per sprint), but was also significantly better at resolving defects (~4.5 per sprint), mitigating risks (~2 per sprint), and resolving technical debt (~1.5 per sprint) than before implementing SAFe. This shows that after adopting SAFe, the team applies a more comprehensive and balanced approach in managing various types of software development work, no longer focusing solely on feature completion.

This increase in Flow Velocity can be attributed to key practices in SAFe that improve coordination and planning. One of the main factors is the implementation of a structured Program Increment Planning (PI Planning). Through PI Planning, team priorities become more aligned, the division of work becomes clear, and the team has the same vision of the target of each increment. As a result, work can be allocated more appropriately and obstacles (such as work overlap or neglect of non-feature aspects) can be identified from the start. Thus, team productivity increases overall - the team can complete more items per sprint - and the completion of work becomes more focused and balanced between features and supporting work (defects, risks, technical debt).

Flow Efficiency

Flow Efficiency is the ratio of value-added time spent on an item to its total completion time (including waiting or blocked time). This metric is expressed as a percentage and indicates how efficient the team's workflow is: the higher the percentage, the greater the portion of time actually used for productive work compared to downtime. Table 2 below presents Flow Efficiency data before and after the implementation of SAFe for the two observed sprints.

Table 3.
Flow Efficiency Before and After Implementation (per Sprint)

Sprint	Value-Adding Time (days)	Waiting Time (days)	Total Time (days)	Flow Efficiency (%)
Program Increment 1 (Before Implementation)				
Sprint 1	10	8	18	55.60%
Sprint 2	12	9	21	57.10%
Average	11	8.5	19.5	56.40%
Program Increment 2 (After Implementation)				
Sprint 1	15	3	18	83.30%
Sprint 2	17	3	20	85.00%
Average	16	3	19	84.20%

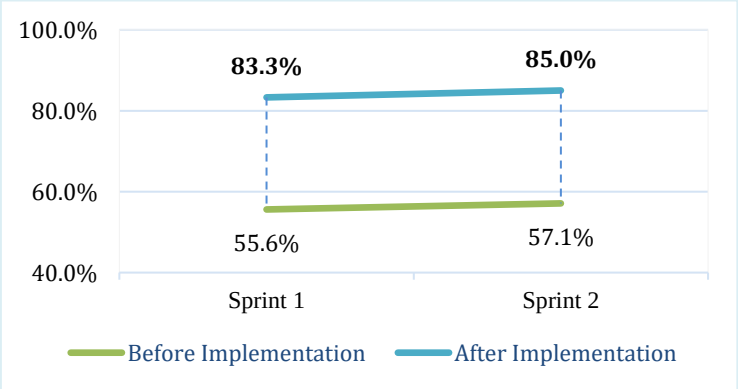

Fig. 4 Flow Efficiency Trends Before and After Implementation

Before the implementation of SAFe, the team had an average Flow Efficiency of around 56.4% (Table 2). This value shows that only approximately half of the total processing time is truly value-added; the rest (almost 44%) is waiting or stopped time. In detail, the team spends an average of 11 days per sprint on value-added work (e.g., coding, testing, etc.), but 8.5 days are wasted in a waiting state (e.g., waiting for approval, waiting for results from other teams, or being held up by something else). The long wait time before SAFe is caused by obstacles such as inter-team dependencies in the backlog and slow approval processes, which cause a lot of work to be stuck in the workflow queue.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Figure 4. is a comparison of Flow Efficiency before and after SAFe implementation (percentage of workflow efficiency). It can be seen that Flow Efficiency increased significantly from an average of 56.4% (Sprint 1: 55.6%, Sprint 2: 57.10%) before SAFe implementation to 84.20% after SAFe implementation (Sprint 1: 83.30%, Sprint 2: 85.00%). This improvement indicates a significant decrease in waiting time and an increase in active time in completing value-added tasks. Before the implementation of SAFe, tasks spent a longer time in a waiting state, which was around 15–17 days per sprint, while after the implementation of SAFe, the waiting time decreased dramatically to around 3 days per sprint. This change indicates that the implementation of SAFe effectively shifts the team's focus to productive activities, thus significantly increasing overall workflow efficiency.

This increase in Flow Efficiency was achieved thanks to the process improvements brought about by SAFe. The application of the Agile Release Train (ART) at the program level improves coordination between teams, so that dependencies can be better managed and synchronization of backlogs between teams is achieved. In addition, process automation through Continuous Integration/Continuous Deployment (CI/CD) also reduces waiting times, for example by speeding up testing and deployment so that teams are not held up waiting for manual processes. This combination of backlog synchronization and automation directly reduces waiting times in the process, which in turn drives the team's Flow Efficiency up significantly after SAFe adoption.

Flow Time

Flow Time is defined as the total time required to complete an item of work from start to finish (end-to-end). This metric covers the entire duration from start to finish, including the processing time and waiting time in between. Flow Time is an important indicator of time-to-market speed or development process responsiveness, because a shorter Flow Time indicates that tasks can be completed faster. Table 3 below shows a comparison of Flow Time for several features (Features 1–3) in the two sprints before and after SAFe implementation, along with their average completion times.

Table 4.
Flow Time Before and After Implementation (in days)

Sprint	Feature 1 (days)	Feature 2 (days)	Feature 3 (days)	Average Flow Time (days)
Program Increment 1 (Before Implementation)				
Sprint 1	15	12	18	15
Sprint 2	14	16	17	15.7
Average	14.5	14	17.5	15.4
Program Increment 2 (After Implementation)				
Sprint 1	8	9	7	8
Sprint 2	10	8	9	9
Average	9	8.5	8	8.5

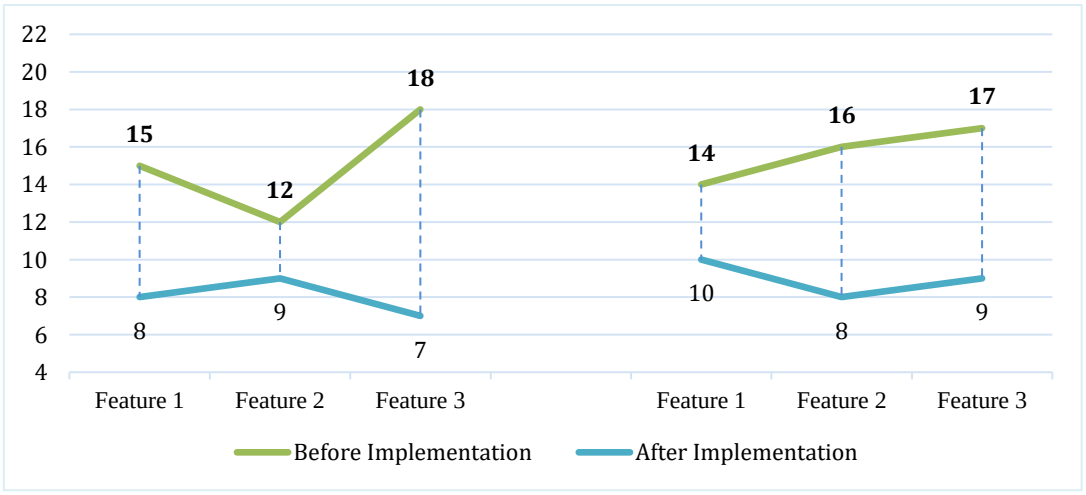

Fig. 5 Flow Time Trends Before and After Implementation (days)

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

The results in Table 3 show a decrease in the average Flow Time per feature from about 15.4 days to 8.5 days after SAFe implementation. Before SAFe, each feature took more than two weeks to complete (an average of about 2 weeks and 1 day). There were variations between features, for example in Sprint 1 before SAFe: Feature 1 was completed in 15 days, Feature 2 in 12 days, and Feature 3 in 18 days. This high flow time was mainly due to the inefficient process before SAFe - there were bottlenecks such as multiple approval procedures, dependencies between teams that hindered the smooth flow of tasks, and a backlog of items waiting to be worked on. These factors caused tasks to take a long time from start to finish.

Figure 5. ia an Average Flow Time before and after SAFe implementation. It can be seen that the average task completion time was reduced by almost half after implementing SAFe (from ~15.4 days to ~8.5 days per item). This decrease in Flow Time means that the team can complete features and other work items faster than before. For example, after SAFe was implemented, a feature that previously took more than 2 weeks could be completed in less than 1 week (8-9 days). This increase in speed of completion reflects a significant improvement in the efficiency of the development process.

Some of the SAFe practices that contributed to lowering Flow Time include: synchronizing the backlog through PI Planning, automating workflows with DevOps/CI-CD, and identifying and mitigating risks early in the cycle. Synchronizing the backlog early in the Program Increment ensures that cross-team dependencies and needs are agreed upon early on, reducing delays throughout the process. The implementation of DevOps and CI/CD accelerates the integration and testing stages, eliminating wait times that previously existed in manual processes (e.g., waiting for development or deployment). In addition, by identifying risks early on and addressing them more proactively, teams can prevent bottlenecks in the middle of feature development. The combination of these initiatives makes for a smoother and faster workflow, which is reflected in the significantly lower Flow Time metric after implementing SAFe.

Flow Load

Flow Load represents the number of work items that the team is actively working on (work in progress/WIP) at a given time or in a sprint. This metric includes all types of items (features, defects, risks, technical debts) that are being processed in parallel. Flow Load is important to observe in order to understand the team's workload and potential bottlenecks: Too high a Flow Load can indicate an overloaded team, risking a decrease in focus and speed of completion. Table 4 below shows the average Flow Load per sprint before and after SAFe adoption, including a breakdown of the number of each active item category.

Table 5.
Flow Load Before and After SAFe Implementation (Number of Active Items per Sprint)

Sprint	Features (Active)	Defects (Active)	Risks (Active)	Debts (Active)	Total Flow Load
Program Increment 1 (Before Implementation)					
Sprint 1	8	5	2	3	18
Sprint 2	7	6	3	4	20
Average	7.5	5.5	2.5	3.5	**19**
Program Increment 2 (After Implementation)					
Sprint 1	5	3	1	2	11
Sprint 2	6	2	1	3	12
Average	5.5	2.5	1	2.5	**11.5**

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14545
e-ISSN : 2541-2019
p-ISSN : 2541-044X

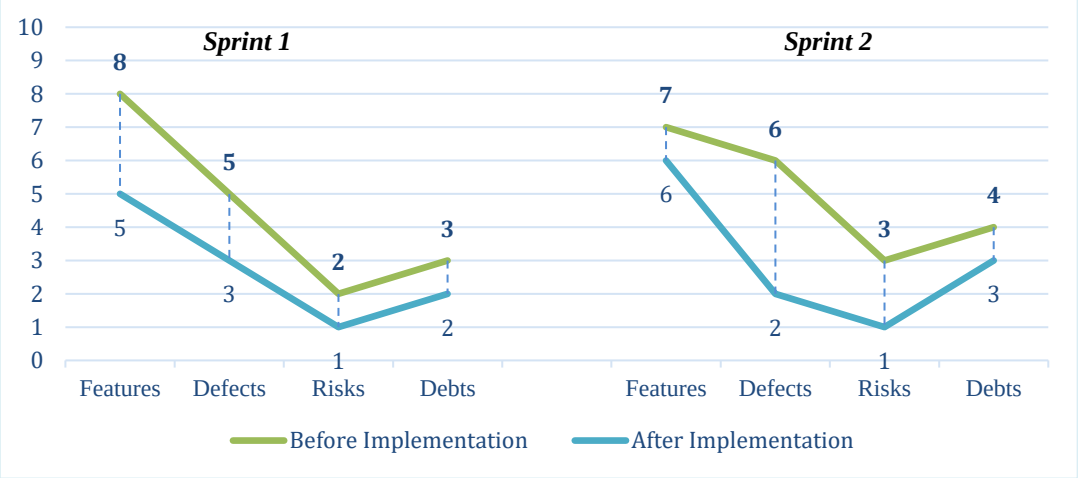

Fig. 6 Active Flow Load Trends Before and After Implementation

Before the implementation of SAFe, the average flow load of the team reached 19 active items per sprint (Table 4). This means that in one sprint the team works on almost 20 items at the same time. In Sprint 1 before SAFe, there were 18 active items, and in Sprint 2 this increased to 20 active items. This high workload indicates excessive multitasking that has the potential to overwhelm the team. As a consequence, tasks tend to be delayed in completion because team members have to divide their focus among too many items at the same time. In addition, before SAFe, there were no strict Work-In-Progress (WIP) limits; as a result, the distribution of the workload among team members was likely to be uneven, and the team had difficulty concentrating its efforts on completing items quickly one by one.

Figure 4. is a comparison of average Flow Load per sprint before and after SAFe. There was a significant decrease in the total number of active items after the implementation of SAFe (from an average of ~19 items to ~11.5 active items worked on per sprint). This decrease in Flow Load indicates that the workload of the team is more manageable post-SAFe, where the team reduces the number of items worked on in parallel. From Figure 4, it can also be seen that the number of each work category (features, defects, risks, technical debt) that is worked on simultaneously decreases after SAFe. For example, the average number of active features decreased from 7.5 to 5.5, and active defects decreased from 5.5 to 2.5. The reduction in WIP in each of these categories indicates that the team has succeeded in avoiding an excessive backlog of work at any one time.

The implementation of SAFe brought about changes in the management of the team's workload, which explains the decrease in Flow Load. The team began to implement Kanban practices with clear limits on WIP to control the amount of parallel work. By limiting WIP, the team ensures that they do not take on new work in excess of capacity before existing work is completed. This prevents overloading and helps distribute work more evenly among team members. As a result, the team experiences less stress and can focus more on completing each item completely before moving on to the next item. This decrease in flow load contributes to increased workflow efficiency: with fewer active tasks, the completion time for each item can be accelerated and the quality of completion is maintained because the team's focus is not divided too much.

Tentative Conclusion

The overall results show that implementing SAFe has a positive impact on four key metrics in software development. The comparison is summarized as follows:

1. Flow Velocity: More than doubled, from an average of 8 items per sprint to around 16.5 items per sprint after SAFe (an increase of ~106% in team output per sprint).
2. Flow Efficiency: Rose dramatically from ~56% to ~84%, reflecting a much higher proportion of productive time and a reduction in lead time of around 5.5 days per sprint.
3. Flow Time: Decreased by almost 50%, from around 15.4 days to 8.5 days on average to complete one item, which means that the time to deliver value to users is much faster.
4. Flow Load: Decreased by around 40%, from an average of 19 active items to 11.5 active items per sprint, indicating a more controlled workload and better team focus on task completion.

DISCUSSIONS

This study aims to evaluate the impact of the implementation of the Scaled Agile Framework (SAFe) on the performance of software development teams using Flow Metrics, namely Flow Velocity, Flow Efficiency, Flow Time, and Flow Load. The results obtained from this study directly answer the research questions and clearly

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

fulfill the original research objectives by showing a significant improvement in the four metrics after the implementation of SAFe based on the calculations in table below:

Table 6.
Improvement of Flow Metrics Before and After SAFe Adoption

Metrics	Average Before SAFe Adoption	Average After SAFe Adoption	Improvement (%)
Flow Velocity (feature)	8	16.5	$\left(\frac{16.5-8}{8}\right) \times 100\% = \mathbf{106.25\%}$
Flow Efficiency (%)	56.4	84.2	$\left(\frac{84.2-56.4}{56.4}\right) \times 100\% = \mathbf{49.33\%}$
Flow Time (days)	15.4	8.5	$\left(\frac{8.5-15.4}{15.4}\right) \times 100\% = \mathbf{-44.81\%}$
Flow Load (feature)	19	11.5	$\left(\frac{11.5-9}{9}\right) \times 100\% = \mathbf{-39.47\%}$

Flow Velocity, as a team productivity indicator measured by the number of features completed per sprint, increased substantially after SAFe implementation. Before the implementation of SAFe, the average team was only able to complete around 8 features per sprint, but this increased dramatically to 16.5 features per sprint after SAFe was implemented (an increase of around 106%). This finding clearly answers the research objective which emphasizes the importance of optimizing the productivity of the software development team through the implementation of SAFe. SAFe practices, such as coordinated planning through the Increment Planning Program (PI Planning) and effective backlog management, have proven to be key factors in significantly increasing Flow Velocity.

The increase in Flow Efficiency is also directly relevant to the research question of how SAFe affects the efficiency of the software development process. Data prior to SAFe implementation showed that team flow efficiency was only 56.4%, indicating a significant amount of waiting time in the work process. After SAFe was adopted, the efficiency increased to 84.2%, or an increase of around 49%. This increase in efficiency is clearly reflected in the research data (see Flow Efficiency trend graph), and confirms that the effective implementation of SAFe is able to reduce wasted waiting time and increase value-added activities in the work process. This is in line with the Lean principles that form the foundation of SAFe, namely the reduction of waste to generate value quickly and efficiently.

In addition, Flow Time shows the relevance of the results to the research objectives, especially related to increasing responsiveness and speed of feature delivery. The research data shows a significant decrease in feature completion time, from an average of 15.4 days before the implementation of SAFe to 8.5 days after the implementation of SAFe (a decrease of around 44.8%). This decrease is a direct impact of the increased work process efficiency and reduced waiting time resulting from SAFe implementation, such as backlog synchronization and Continuous Integration/Continuous Deployment (CI/CD) automation. With shorter Flow Time, teams become more responsive to changing business needs and are able to provide added value to users at a faster pace.

Furthermore, the Flow Load also shows relevant results that address the research objective of improving the effectiveness of team workload management. Prior to SAFe implementation, the average team handled about 19 active features simultaneously per sprint. After the implementation of SAFe, this number decreased substantially to 11.5 active features per sprint (a decrease of around 39%). This data shows that the implementation of SAFe explicitly helps teams manage capacity and work priorities through Work-In-Progress (WIP) restrictions, reducing multitasking and increasing focus and the quality of the team's overall work output.

Overall, the results of this study explicitly show a clear relationship between the data obtained and the questions and research objectives set out previously. These findings collectively show that the implementation of SAFe makes a real contribution to optimizing productivity, efficiency, and workflow management of software development.

From a practical point of view, the results of this study have important implications for organizations and software development teams that want to implement SAFe. The clearly presented empirical data shows that using SAFe doesn't just change the way teams work in theory, but actually has a real measurable impact on team output. For example, increased Flow Velocity allows teams to deliver features to market faster, increasing product competitiveness and customer satisfaction. Increased Flow Efficiency means that the team uses time more optimally by reducing wasted waiting time, while decreased Flow Load creates a healthier and more focused work environment with a balanced workload.

Based on the results of this study, here are practical recommendations for companies that want to implement SAFe and use Flow Metrics as a tool to evaluate the effectiveness of the implementation:

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

1. Baseline Measurement and Continuous Monitoring: Set a Flow Metrics baseline before implementing SAFe, then monitor it regularly and continuously to evaluate the effectiveness of the implementation.
2. Work-In-Progress (WIP) Restrictions: Apply clear WIP limits to ensure that the team's workload remains balanced and the focus of work is maintained.
3. Workflow Automation: Maximize the use of automation such as CI/CD to reduce waiting time and increase Flow Efficiency.
4. Realistic Capacity Planning: Ensure that work planning is based on actual team capacity, taking into account clear business priorities to maintain Flow Velocity and Flow Load stability.

CONCLUSION

This study shows that the implementation of the Scaled Agile Framework (SAFe), with the use of Flow Metrics as a measuring tool, has clearly succeeded in meeting the initial research objective of increasing efficiency and productivity in software development. The use of Flow Metrics enables teams to effectively identify bottlenecks, simplify work processes, and increase the value provided to customers. This is evidenced by a significant increase in Flow Velocity and Flow Efficiency, as well as a decrease in Flow Time and Flow Load after the implementation of SAFe. In particular, an increase in Flow Velocity of 106.25%, an increase in Flow Efficiency of up to 49.33%, and a decrease in Flow Time of 44.81% and Flow Load of 39.47%, reflect real achievements towards the goal of increasing team efficiency and productivity in the software development process.

Success in achieving these goals is mainly due to the implementation of SAFe's core components, including backlog synchronization, Continuous Integration/Continuous Deployment (CI/CD) automation, and the implementation of the Increment Planning (PI Planning) Program. Through these components, organizations can effectively improve team collaboration, respond to rapidly changing business needs, and improve overall operational efficiency.

This study also provides practical recommendations for companies that want to implement SAFe as a large-scale software development framework. The recommendations include: start the implementation with a pilot project to identify challenges early, prioritize PI Planning to align business needs with team capacity, use Flow Metrics as the main technique to monitor team performance regularly, increase automation of development processes through CI/CD to accelerate value delivery, and optimize collaboration through backlog synchronization and a good communication plan. In addition, regular retrospective meetings and coaching on SAFe and Flow Metrics principles and tools are also important to ensure the effectiveness and long-term sustainability of the implementation.

Thus, the conclusion of this study explicitly shows that the utilization of Flow Metrics in the context of SAFe implementation has succeeded in meeting the research objectives in improving the efficiency and productivity of software development teams.

REFERENCES

Abou-Aly, A., Khodeir, L., & ElAbd, N. (2024). Identification of the Need for Agile Methodologies in Construction Projects Design Management. *Fayoum University Journal of Engineering*, *7*(2), 152–166. https://doi.org/10.21608/FUJE.2024.343809

Agile Alliance. (n.d.). Retrieved January 2, 2025, from https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/

Amajuoyi, P., Benjamin, L. B., Adeus, K. B., Amajuoyi, P., Benjamin, L. B., & Adeus, K. B. (2024). Agile Methodologies: Adapting Product Management to Rapidly Changing Market Conditions. *Global Scholarly Communication Online Press*, *19*(2), 249–267. https://doi.org/10.30574/GSCARR.2024.19.2.0181

Bakar, S. A., & Dorasamy, M. (2023). From Adoption to Sustainability: A Journey of Large-Scale Agile Implementation. *International Journal of Technology*, *14*(6), 1367–1379. https://doi.org/10.14716/IJTECH.V14I6.6645

Chahal, S. (2023). Agile Methodologies for Improved Product Management. *Journal of Business and Strategic Management*, *8*(4), 79–94. https://doi.org/10.47941/JBSM.1439

Cvejič, M., & Cvejič, M. (2022). Evolution of Agile Practices During a M & A of a European and a Chinese Company. *Open Journal of Business and Management*, *10*(5), 2378–2388. https://doi.org/10.4236/OJBM.2022.105118

Edison, H., Wang, X., & Conboy, K. (2022). Comparing Methods for Large-Scale Agile Software Development: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, *48*(8), 2709–2731. https://doi.org/10.1109/TSE.2021.3069039

Flow Framework. (n.d.). Retrieved January 10, 2025, from https://flowframework.org/

Gustavsson, T., Berntzen, M., & Stray, V. (2022). Changes to Team Autonomy in Large-Scale Software Development: A Multiple Case Study of Scaled Agile Framework (SAFe) Implementations. *International*

*name of corresponding author

Journal of Information Systems and Project Management, *10*(1), 29–46. https://doi.org/10.12821/ijispm100102

Nuottila, J., Aaltonen, K., & Kujala, J. (2022). Challenges of Adopting Agile Methods in a Public Organization. *International Journal of Information Systems and Project Management*, *4*(3), 65–85. https://doi.org/10.12821/ijispm040304

Pandey, S. K., Mosin, V., Durisic, D., Koppisetty, A. C., Staron, M., & History, A. (2023). Data Handling for Assuring Production Quality of Image Intensive Autonomous Drive Systems: An Industrial Case Study. *Journal of Software Engineering for Autonomous Systems*, *1*(1–2), 29–47. https://doi.org/10.55060/J.JSEAS.231018.001

Pinciroli, F. (2024). Selection of Agile Project Management Approaches Based On Project Complexity. *Journal of Software: Evolution and Process*, *36*(12), e2716. https://doi.org/10.1002/SMR.2716

Pinto, H. W., Kerzner, R. R., & Cleland, J. D. (2023). Exploring the Implementation of Agile Project Management in the United States Construction Industry: Benefits, Challenges, and Success Factors. *Journal of Entrepreneurship & Project Management*, *7*(7), 11–23. https://doi.org/10.53819/81018102T4163

Project Management Institute. (2021). *A Guide to the Project Management Body of Knowledge (PMBOK® Guide) - Seventh Edition and the Standard for Project Management*.

Scaled Agile Framework. (n.d.). Retrieved January 5, 2025, from https://framework.scaledagile.com/

Wijaya, S. F., Wijaya, S. F., & Egeten, A. E. J. (2019). Breaking Through Unravel Problems in ERP Implementation Using Agile. *Journal of Advanced Computer Science & Technology (JACST)*, *8*(2), 16–22. https://doi.org/10.14419/jacst.v8i2.16236

*name of corresponding author