<	...
Online team meeting	https://fau.zoom-x.de/j/9776061710?pwd=9BPbcHYQaVEf6L0IH3xbsSeNzajvJ0.1
Production system (if any)	...
Test system (if any)	...
GitHub repository	https://github.com/amosproj/amos2024ws03-android-zero-instrumentation
GitHub feature board	https://github.com/orgs/amosproj/projects/72/views/2
GitHub imp-squared backlog	https://github.com/orgs/amosproj/projects/76
Team T-shirt (white)	https://www.shirtinator.de/s/OaDrwZ0JQ9WL1QrhmOU7KA
Team T-shirt (black)	https://www.shirtinator.de/s/Ou9CCXOBQIW04aOC_Hov6g
Additional materials	...
Team maling list	oss-amos-proj3@lists.fau.de

Last Name	First Name	GitHub User Name	Email Address
Krug	Maximilian	HaruspexSan	krugm03@gmail.com
Ayach	Mohammed Tamim	Tamemo99	Tamemayash@gmail.com / Ayachmoh@hu
Bretting	Luca	luca-dot-sh	luca.bretting@fau.de
Seidl	Robin	mr-kanister	robin.seidl@fau.de (main) / 68117355+Mr-k
Labroussis	Christos	clabrous	c.labroussis1@gmail.com
Hilgers	Felix	fhilgers	felix.hilgers@fau.de
Weisshuhn	Tom	der-whity	tom.weisshuhn@fau.de
Schlicht	Franz	ffranzgitHub	franz.schlicht@fau.de
Nawlo	Ali	alinawlo	ali.nawlo@campus.tu-berlin.de
Zinn	Benedikt	BenediktZinn	benedikt.wh.zinn@gmail.com

#	Meeting Day	Product Owners	Software Developer	Release Manager	Scrum Master	Comment
1	2022-10-16	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	
2	2022-10-23	Ali Nawlo	Everyone else	Maximlian Krug	Maximilian Krug	
3	2022-10-30	Mohammed Tamim Ayach	Everyone else	Benedikt Zinn	Maximilian Krug	
4	2022-11-06	Ali Nawlo	Everyone else	Tom Weißhuhn	Maximilian Krug	
5	2022-11-13	Mohammed Tamim Ayach	Everyone else	Robin Seidl	Maximilian Krug	
6	2022-11-20	Ali Nawlo	Everyone else	Franz Schlicht	Maximilian Krug	
7	2022-11-27	Mohammed Tamim Ayach	Everyone else	Benedikt Zinn	Maximilian Krug	Mid-term due
8	2022-12-04	Ali Nawlo	Everyone else	Robin Seidl	Maximilian Krug	
9	2022-12-11	Mohammed Tamim Ayach	Everyone else	Luca Bretting	Maximilian Krug	
10	2023-01-11	Ali Nawlo	Everyone else	Tom Weißhuhn	Maximilian Krug	
11	2023-01-18	Mohammed Tamim Ayach	Everyone else	Felix Hilgers	Maximilian Krug	
12	2023-01-25	Ali Nawlo	Everyone else		Maximilian Krug	
13	2023-02-01	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	
14	2023-02-08	Ali Nawlo	Everyone else		Maximilian Krug	Demo day!
15	2023-02-15	Mohammed Tamim Ayach	Everyone else		Maximilian Krug	Retrospective
Product owners, software developers, and Scurm Master are set and ideally don't change over time; the critical part is the Release Manager role you need to define here						

Goals 1	
	Completing the objective and task given by our IP, becoming a well rounded team in the meantime
Meeting norms 2	Be punctual (with a 5min pardon time)
	Max. two times missing from IP meeting
	not having the camera off two consecutive times
Working norms 2	Don't push to main, keep main in working order
	Dependencies are a team effort
	all tests must pass
	criticism via pull/merge requests
Coordination norms 2	PR with one other member
Communication norms 2	communication via discord - team meeting via zoom
	document major changes
Consideration norms 2	be repectfull
	small disagreement, discuss and vote
Cont. improvement norms 2	team meeting for tracking team's progress -> standup emails for gathering intel
	pushing non functional changes will trigger a workshop
Rewards 1	have cake together
Sanctions 1	Otheres choose a random virtual background
Signatures	
Scrum Master	Maximilian Krug
Product owner	Mohammed Tamim Ayach
Product owner	Ali Nawlo
Software developer	Luca Bretting
Software developer	Benedikt Zinn
Software developer	Christos Labroussis
Software developer	Robin Seidl
Software developer	Franz Schlicht
Software developer	Felix Hilgers
Software developer	Tom Weißhuhn
	https://oss.cs.fau.de/wp-content/uploads/2014/04/Team-Contract-Explanation-and-Examples.pdf

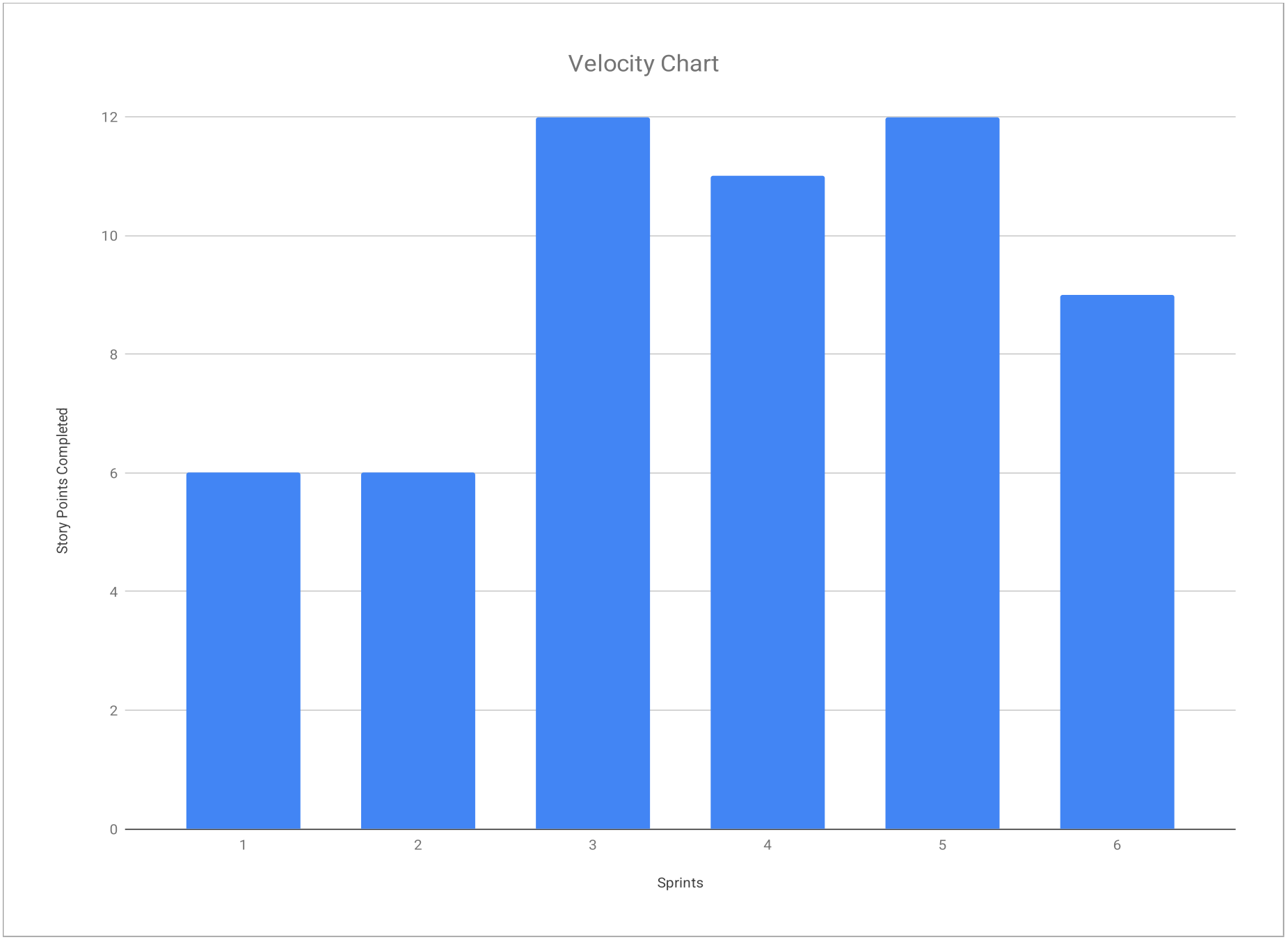
Product Vision	Project Mission
In systems with a high frequency of component changes, it is difficult to determine which component might be causing performance issues and affecting the entire system negatively. This is especially hard if the source code and/or build environment for the components is not present as they might be coming from external suppliers, which means they cannot easily be instrumented. This can result in a lot of communication and extra work. Using eBPF allows for tracking some of these issues at the kernel level, where for example blocking calls are made and can be tracked. It allows for hooking into Sys-Calls as well as calls to other userspace or kernel-level functions (uprobes and kprobes), all without needing to modify application code. This makes it possible to track down cross-cutting performance issues without needing additional support from the vendor of the component. The information about, for example the length of a blocking calls, can then be passed to various frontends, such as an Android application running on the target hardware or an external sink for displaying the data in visualization software like Grafana.	ZIOFA (Zero Instrumentation Observability for Android) aims to implement observability use cases relevant to performance specified by our industry partner using eBPF. Examples include tracing long-running blocking calls, leaking JNI indirect references or signals like SIGKILL sent to processes, all without instrumenting the observed application itself. The eBPF programs are loaded and unloaded using a backend daemon running as root that will collect metrics and send them to a client. For displaying these metrics to the user, we are implementing an on-device UI that can display visualizations for these use cases and allow for configuration of the enabled use cases, but using a decoupled Client SDK so that future work may easily make the data accessible the external processing.

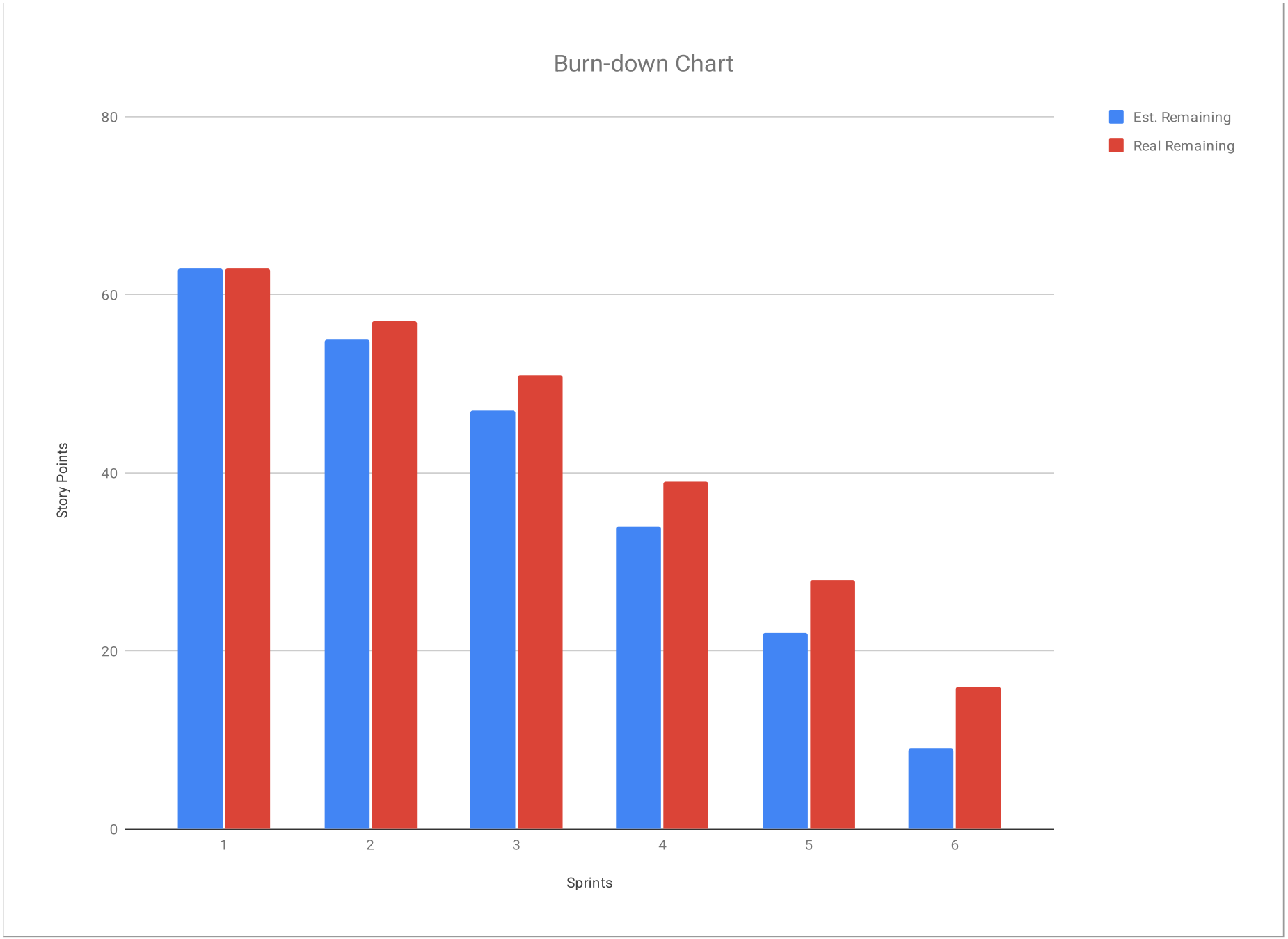
Term	Definition

Sprint #	Sprint goal
1	None
2	None
3	None
4	Optional
5	Working, loading, and unloading of eBPF Programs from UI all the way to eBPF
6	Analyzing traffic over Unix Domain Sockets
7	Analyzing user space function calls
8	Finalizing User space function calls
9	Improve Testing and Finalize previous work
10	Implementing SIGQUIT, finalzing testing, and starting Database
11	Garbage Collection
12	Implementing File Descriptors, finalzing older tasks
13	Bugfixing and refactoring / finishing smaller issues
14	
15	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
Release						
Total			63	63		
Sprints						
1	**Get to know the Team**		8	63	6	63
2	**Get familiar with eBPF and other required technologies.**		8	55	6	57
3	**Start Developing, have a UI blueprint and a Backend beginning**		13	47	12	51
4	**Build a UI and work with Ebpf**		12	34	11	39
5	**Working, loading, and unloading of eBPF Programs from UI all the way to eBPF**		13	22	12	28
6	**Analysing traffic over Unix Domain Socket**		9	9	9	16
Features						
1	**Get to know the Team**					
		Brain-storm Architecture	3		1	
		Preperation of Kotlin	3		3	
		Brain-storm ebpf use cases	2		2	
2	**Get familiar with eBPF and other required technologies.**					
		Docker Container	3		3	
		get information about android processes to list them	3		1	
		set aarch64 als target	1		1	
		use android 13 instead of 15	1		1	
3	**Start Developing, have a UI blueprint and a Backend beginning**					
		Preparation of CI	3		3	
		find timeseries visualization library	2		2	
		Sbom generation	2		1	
		Generation of sboms doesn't include kotlin	1		1	
		Communcation between Android side and Rust side	5		5	
4	**Build a UI and work with Ebpf**					
		unix domain socket traffic analysis (research)	5		3	
		Home Screen and Navigation Drawer	2		3	
		EBPF Program extension to load kProbes	3		3	
		Implement frontend load and list programs	2		2	
5	**Working, loading, and unloading of eBPF Programs from UI all the way to eBPF**					
		kotlin interface for frontend loading and listing programs	1		1	
		test cli client: load and list programs	3		2	
		client library exported to kotlin	2		1	
		Running processes List	3		3	
		loading/unloading of ebpf functions in daemon	2		3	
		Display Installed Procceses in UI	2		2	
6	**Analysing traffic over Unix Domain Socket**					
		collecting unix domain sockets events	2		2	
		configuring unix domain socket tracing	2		2	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		visualizing traffic from unix domain socket	3		3	
		setting tracepoint for sendmsg syscalls	2		2	


Velocity Chart
12
10
8
6
4
2
0
Story Points Completed
1
2
3
4
5
6
Sprints


Burn-down Chart
Est. Remaining
Real Remaining
Story Points
80
60
40
20
0
1
2
3
4
5
6
Sprints

White Paper: Standardizing the Peer Review Process

Introduction

In the fast-paced world of software development, a robust peer review process is essential to maintain code quality and ensure successful project outcomes. This white paper outlines best practices and guidelines to standardize the peer review process, fostering a collaborative and efficient development environment.

Key Components of an Effective Peer Review Process

1. Clear Communication and Understanding

- **Technical Clarity**: Reviewers must have a clear understanding of the technical goals and requirements of the code being reviewed. This ensures that feedback is relevant and constructive.
- **Effective Communication Channels**: Utilize various communication mediums beyond GitHub comments, such as video calls or chat platforms, to discuss complex issues or clarifications.

2. Commit and Pull Request (PR) Standards

- **Descriptive Commits**: Each commit should have a clear, descriptive message explaining its

- **Descriptive Commits**: Each commit should have a clear, descriptive message explaining its purpose and changes. This helps reviewers understand the progression of the code.
- **Short Usage Descriptions in PRs**: Provide a concise description of how the code changes can be tested or used, aiding the reviewer in understanding the context and functionality.

3. Code Quality and Structure

- **Consistent Naming Conventions**: Adhere to established naming conventions and code structures to make the codebase easier to navigate and review.
- **Avoiding Nitpicking**: Focus on substantial issues rather than minor spelling errors in comments, unless they impact code functionality or clarity.

4. Interactive Review Process

- **Hands-On Code Interaction**: Reviewers should check out the branch locally and interact with the code to gain a deeper understanding, rather than solely relying on the GitHub interface.
- **Pair Programming**: For complex issues, consider pair programming sessions to collaboratively solve problems and enhance understanding.

5. Reviewer Responsibilities

- **No Superficial Approvals**: Avoid approving code with "Looks Good To Me" (LGTM) without a thorough understanding of the changes.
- **Constructive Feedback**: Provide actionable, constructive feedback that helps the author improve the code and learn from the review process.

Conclusion

By standardizing the peer review process with these guidelines, development teams can enhance code quality, foster collaboration, and ensure that all team members are aligned with project goals. Implementing these practices will lead to more efficient development cycles and a more cohesive team dynamic.

Standardizing the Pair Programming Process: A White Paper

Introduction

Pair programming is a collaborative approach to software development where two programmers work together at one workstation. This method not only enhances code quality but also facilitates knowledge sharing and team cohesion. This white paper aims to standardize the pair programming process by integrating best practices and strategies to maximize its effectiveness.

Key Principles of Pair Programming

1. **Role Switching**:
   - Regularly switch roles between the driver (the one writing the code) and the navigator (the one reviewing and guiding). This ensures both participants are equally engaged and can contribute their perspectives.
   - For similar tasks, alternate roles frequently to maintain focus and energy.
2. **Effective Communication**:
   - Use visualization tools, such as whiteboards, to explain complex ideas or concepts. This is particularly useful if one participant struggles to understand a concept.

- Share plans and strategies for solving tasks with your partner to incorporate diverse ideas and approaches.

3. **Collaborative Learning**:
   - Target programmers who work on unfamiliar areas to broaden understanding and skills.
   - When experimenting with new tools, like the oatdump program, switch screen sharing to ensure both participants gain hands-on experience.
4. **Code Review and Troubleshooting**:
   - Utilize shared pull requests to discuss and explain code changes. This practice helps in understanding the rationale behind code decisions and fosters collective code ownership.
   - Maintain a wiki for common troubleshooting issues and code comments to streamline problem-solving.
5. **Checkpoints and Feedback**:
   - Establish "checkpoints" during pair programming sessions. After completing a significant portion of code, the navigator should run the current state on their device to verify understanding and functionality.
   - Encourage open feedback and discussions to continuously improve the pair programming process.
6. **Mob Programming for Architectural Changes**:
   - For significant architectural changes, consider mob programming, where the entire team collaborates on the code. This approach ensures that all team members are aligned with the changes and can contribute their insights.

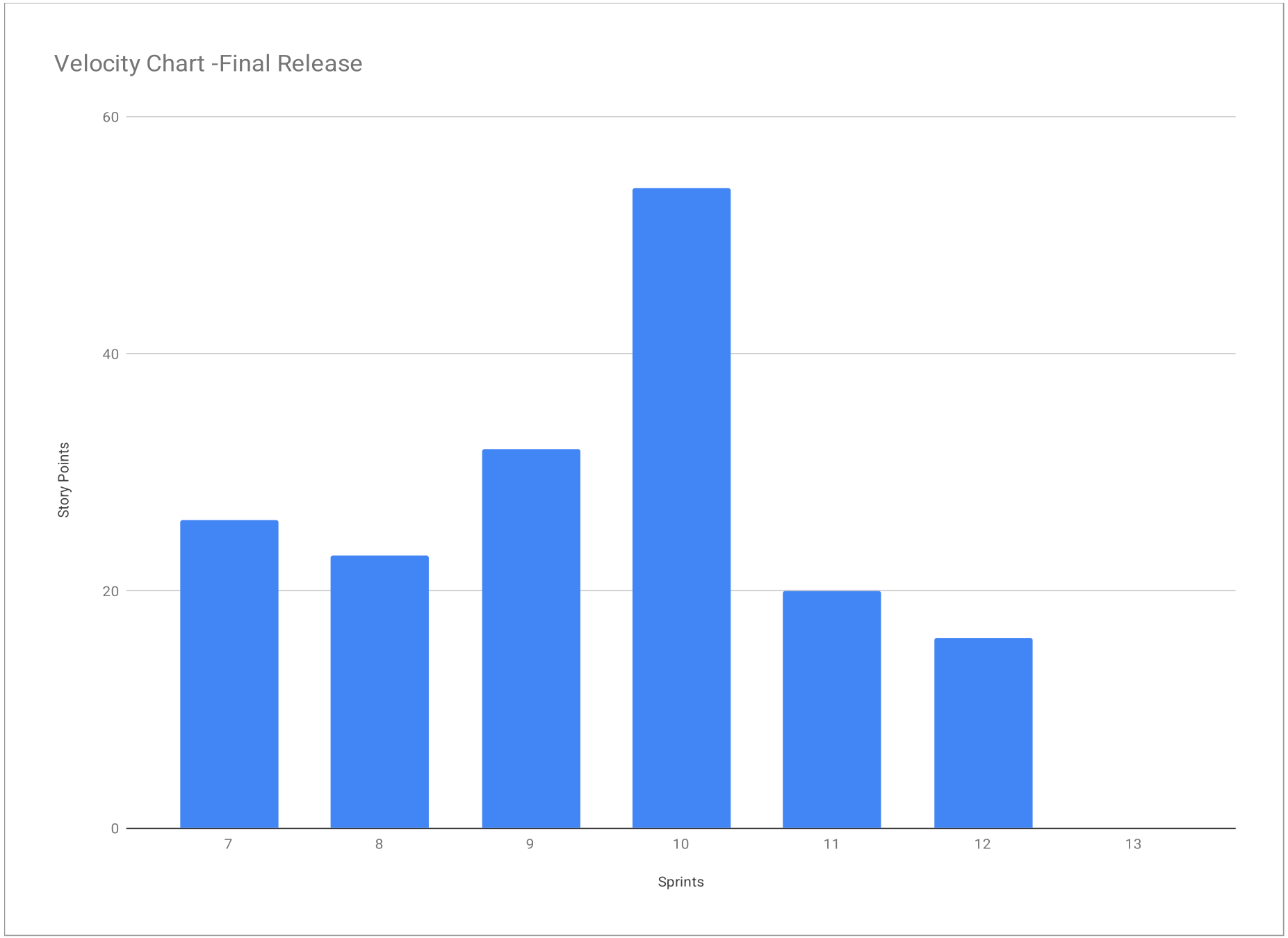
Conclusion

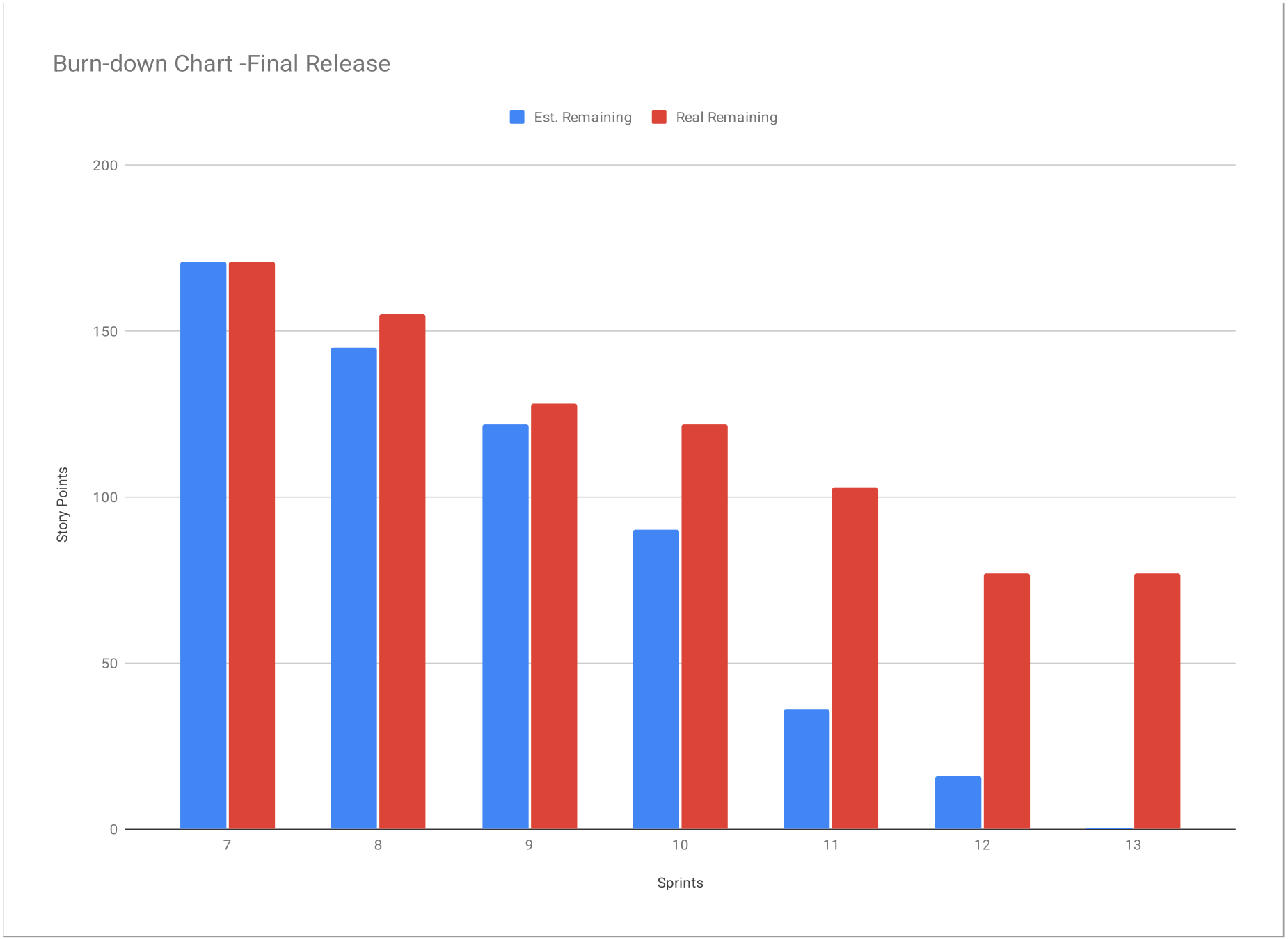
By standardizing the pair programming process through these practices, teams can enhance collaboration, improve code quality, and foster a culture of continuous learning and improvement. Implementing these strategies will not only streamline the development process but also create a more cohesive and knowledgeable team.

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
Release						
Total			171	171		
Sprints						
7	**Analyzing user space function calls**		26	171	16	171
8	**Finalizing User Space Function Calls**		23	145	27	155
9	**Improve Testing and finalize previous work**		32	122	6	128
10	**Implementing SIGQUIT, finalzing testing, and starting Database**		54	90	19	122
11	**Garbage Collection**		20	36	26	103
12	**Implementing File Descriptors, finalzing older tasks**		16	16	0	77
13				0		77
Features						
7	Analyzing user space function calls					
		setup Uprobe Analysis	2		5	
		Configure Uprobe Analysis	1		-	
		Uprobe Events in Frontend	2		-	
		Collect Uprobe events	1		-	
		defining metrics for the visualization screen	3		3	
		setup ebpf uprobes	3		-	
		Dex/Oat Symbols	5		-	
		Project Refactored	9		8	
8	Finalizing User Space Function Calls					
		Configure Uprobe Analysis	2		2	
		Uprobe Events in Frontend	1		-	
		Collect Uprobe events	1		3	
		setup ebpf uprobes	3		2	
		Dex/Oat Symbols	5		8	
		Refactoring ebpf Programs	1		1	
		Refactoring Configuration API	2		2	
		Refactoring pIDs to uint32	2		1	
		Refactoring Collection of events in Daemon	2		5	
		Uprobe Analysis: Frontend Show Symbols	2		3	
		Uprobe Analysis: Frontend Show Uprobe Events	2		-	
9	Improve Testing and finalize previous work					
		Search bar to filter out App/Process	2		2	
		Aggregate Data Points in Background for efficient processing	2		-	

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		Uprobe Analysis: Frontend Show Uprobe Events	2		-	
		Mocking IO in userspace daemon	3		-	
		Integration Testing	3		2	
		Integration Testing 2	5		-	
		Ebpf Programs testing	3		-	
		Uprobe Analysis: Finding Symbols from Shared Libraries	2		-	
		In Memory testing	2		-	
		Ebpf: SIGQUIT	2		2	
10	Implementing SIGQUIT, finalzing testing, and starting Database					
		Refactor: SIGQUIT	1		1	
		Daemon: config SIGQUIT	1		1	
		Collector: for information resulting from SIGQUIT calls	1		1	
		Frontend: SIGQUIT	2		2	
		Integration Testing 2	5		5	
		Prototype for overlay mode	3		-	
		Visualize JNI Reference Metrics	3		2	
		Aggregate Data Points in Background for Efficient Processing	2		-	
		Actor Refactor	1		-	
		In memory testing	2		2	
		Mocking IO in userspace daemon	3		-	
		Uprobe: Trace JNI symbols	3		-	
		Uprobe: expand client library	3		-	
		Testing Ebpf Programs	3		5	
		Uprobe Analysis: Frontend Show Uprobe Events	2		-	
11	Garbage Collection					
		Prototype for overlay mode	3		-	
		Aggregate Data Points in Background for Efficient Processing	2		-	
		Mocking IO in userspace daemon	3		-	
		Uprobe: expand client library	3		-	
		Uprobe: Trace JNI symbols	3		3	
		Bug Fix: Switch to Chart for SIGQUIT	2		2	
		Garbage Collection	5		13	
		EPIC: Create a Databank in the Backend	-1		8	
12	Implementing File Descriptors, finalzing older tasks					
		eBPF: Monitor File Descriptor Usage	2			
		Actor Refactor	1			
		bugfix - Scrolling functionality in the UI	2			
		bugfix - Vizualisation Disappearing	2			
		Daemon: Integrate File Descriptor Monitoring	1			

Sprint	Goal	Feature Name	Est. Size	Est. Remaining	Real Size	Real Remaining
		Prototype for overlay mode	3			
		Aggregate Data Points in Background for Efficient Processing	2			
		Mocking IO in userspace daemon	3			
13						


Velocity Chart -Final Release
60
40
20
0
Story Points
7
8
9
10
11
12
13
Sprints


Burn-down Chart -Final Release
Est. Remaining
Real Remaining
200
150
100
50
0
Story Points
7
8
9
10
11
12
13
Sprints

#	Feature Definition of Done	Sprint Release Definition of Done	Project Release Definition of Done
	1. Code for Components has been written. a. The code does comply to the naming conventions of the used programming language b. Code has been completed c. Unclear code parts are provided with a short comment, to explain what this part is supposed to do. 2. Developers submit a screenshots of the finished feature as a comment to the related issue 3. Feature has been reviewed by another team member 4. Feature has been merged and closed	1. Finished issues are marked as done 2. Code is tested and deployed 3. A short demo is available for each sprint (this is compliant with point 3 in DoD for Feature) so it can be the screenshots or a small video or even a short-live presentation 4. Bill of Material is kept in a current state	1. Team agrees on which features to be released 2. Features have been tested and reviewed by other team member 3. Documentations are kept updated 4. A short demo featuring major features is provided

Type	Link / reference

#	Context	Name	Version	License	Comment
1	Gradle Plugin	org.cyclonedx.bom	1.10.0	APACHE-2.0	https://github.com/CycloneDX/cyclonedx-gradle-plugin
2	Gradle Plugin	nl.littlerobots.version-catalog-update	0.8.5	APACHE-2.0	https://github.com/littlerobots/version-catalog-update-plugin
3	Gradle Plugin	com.github.ben-manes.versions	0.51.0	APACHE-2.0	https://github.com/ben-manes/gradle-versions-plugin
4	Gradle Plugin	com.android.application	8.6.0	APACHE-2.0	https://maven.google.com/web/index.html?q=com.android.applicat#com.android.application:com.android.application.gradle.plugin:8.6.0
5	Gradle Plugin	com.ncorti.ktfmt.gradle	0.21.0	MIT	https://github.com/cortinico/ktfmt-gradle
6	Gradle Plugin	org.jetbrains.kotlin.plugin.compose	2.1.0	APACHE-2.0	https://github.com/JetBrains/compose-multiplatform
7	Gradle Plugin	org.jetbrains.kotlin.android	2.1.0	APACHE-2.0	https://github.com/JetBrains/kotlin
8	Android UI	androidx.activity:activity-compose	1.9.3	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.activity#androidx.activity:activity-compose:1.9.3
9	Android UI	androidx.compose:compose-bom	2024.12.01	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.compose#androidx.compose:compose-bom:2024.12.01
10	Android UI	androidx.core:core-ktx	1.15.0	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.core#androidx.core:core-ktx:1.15.0
11	Android UI	androidx.lifecycle:lifecycle-runtime-ktx	2.8.7	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.life#androidx.lifecycle:lifecycle-runtime-ktx:2.8.7
12	Android DI	io.insert-koin:koin-android	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
13	Android DI	io.insert-koin:koin-androidx-compose	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
14	Android DI	io.insert-koin:koin-core	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
15	Android Test	io.insert-koin:koin-test-junit4	4.0.0	APACHE-2.0	https://github.com/InsertKoinIO/koin
16	Android Test	androidx.test.espresso:espresso-core	3.6.1	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.test.es#androidx.test.espresso:espresso-core:3.6.1
17	Android Test	androidx.test.ext:junit	1.2.1	APACHE-2.0	https://maven.google.com/web/index.html?q=androidx.test.ext#androidx.test.ext:junit:1.2.1
18	Android Test	junit:junit	4.13.2	EPL-1.0	https://github.com/junit-team/junit4
19	Rust Ebpf	aya	0.13.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
20	Rust Ebpf	aya-ebpf	0.1.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
21	Rust Ebpf	aya-log	0.2.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
22	Rust Ebpf	aya-log-ebpf	0.1.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
23	Rust Ebpf	libc	0.2.168	MIT OR APACHE-2.0	https://github.com/rust-lang/libc
24	Rust Errors	anyhow	1.0.0	MIT OR APACHE-2.0	https://github.com/dtolnay/anyhow
25	Rust Build	cargo_metadata	0.19.1	MIT	https://github.com/oli-obk/cargo_metadata
26	Rust Build	clap	4.5.23	MIT OR APACHE-2.0	https://github.com/clap-rs/clap
27	Rust Build	which	7.0.0	MIT	https://github.com/harryfei/which-rs
28	Rust Logging	env-logger	0.11.5	MIT OR APACHE-2.0	https://github.com/rust-cli/env_logger
29	Rust Logging	log	0.4.22	MIT OR APACHE-2.0	https://github.com/rust-lang/log
30	Rust Async	tokio	1.42.0	MIT	https://github.com/tokio-rs/tokio
31	Rust Async	tokio-stream	0.1.17	MIT	https://github.com/tokio-rs/tokio
32	Rust API	prost	0.13.4	APACHE-2.0	https://github.com/tokio-rs/prost
33	Rust API	tonic	0.12.3	MIT	https://github.com/hyperium/tonic
34	Rust API	tonic-build	0.12.3	MIT	https://github.com/hyperium/tonic
35	Toolchain	python3	3.12.6	PSF-2.0	https://docs.python.org/3/license.html
36	Toolchain	rust	1.84.0-nightly	MIT OR APACHE-2.0	https://www.rust-lang.org/policies/licenses
37	Toolchain	cargo-ndk	3.5.7	MIT OR APACHE-2.0	https://github.com/bbqsrc/cargo-ndk
38	Toolchain	protoc	28.2	BSD-3-Clause	https://github.com/protocolbuffers/protobuf
39	Toolchain	bpf-linker	0.9.13	MIT OR APACHE-2.0	https://github.com/aya-rs/bpf-linker
40	Toolchain	nix	2.18.7	LGPL-2.1	https://github.com/NixOS/nix
41	Toolchain	cyclonedx-cli	0.25.1	APACHE-2.0	https://github.com/CycloneDX/cyclonedx-cli
42	Toolchain	gradle	8.10.2	APACHE-2.0	https://github.com/gradle/gradle
43	Toolchain	openjdk	21.0.3	GPL-2.0-with-classpath-exception	https://openjdk.org/legal/gplv2+ce.html
44	Toolchain	android-cmdline-tools	16	android-sdk-license	https://developer.android.com/studio/terms
45	Toolchain	android-emulator	35.3.6.0	android-sdk-license	https://developer.android.com/studio/terms
46	Toolchain	android-ndk	28.0.12433566	android-sdk-license	https://developer.android.com/studio/terms
47	Toolchain	android-tools	35.0.0	android-sdk-license	https://developer.android.com/studio/terms
48	Toolchain	platform-tools	35.0.2	android-sdk-license	https://developer.android.com/studio/terms
49	Toolchain	platforms-android	35	android-sdk-license	https://developer.android.com/studio/terms
50	Rust API	uniffi	0.28.2	MPL-2.0	https://github.com/mozilla/uniffi-rs
51	Rust API	thiserror	2.0.6	MIT OR APACHE-2.0	https://github.com/dtolnay/thiserror
52	Gradle Plugin	com.android.library	8.7.3	APACHE-2.0	https://maven.google.com/web/index.html?q=com.android.libr#com.android.library:com.android.library.gradle.plugin:8.7.3
53	Gradle Plugin	org.mozilla.rust-android-gradle.rust-android	0.9.4	APACHE-2.0	https://github.com/mozilla/rust-android-gradle
54	Android Rust	net.java.dev.jna	5.15.0	Apache-2.0 OR LGPL-2.1	https://github.com/java-native-access/jna
55	Android Navigation	androidx.navigation:navigation-compose	2.8.5	Apache-2.0	https://maven.google.com/web/index.html?q=androidx.navigation#androidx.navigation:navigation-compose:2.8.5
56	Android Logging	com.jakewharton.timber:timber	2.8.0	Apache-2.0	https://github.com/JakeWharton/timber
57	Android Visualization	com.partykandpatrick.vico:compose	2.0.0-beta.3	Apache-2.0	https://github.com/patrykandpatrick/vico
58	Android Visualization	com.partykandpatrick.vico:compose-m2	2.0.0-beta.3	Apache-2.0	https://github.com/patrykandpatrick/vico
59	Android Visualization	com.partykandpatrick.vico:compose-m3	2.0.0-beta.3	Apache-2.0	https://github.com/patrykandpatrick/vico
60	Android Visualization	com.partykandpatrick.vico:core	2.0.0-beta.3	Apache-2.0	https://github.com/patrykandpatrick/vico
61	Rust Serialization	serde	1.0.215	MIT OR APACHE-2.0	https://github.com/serde-rs/serde
62	Rust Serialization	serde-json	1.0.0	MIT OR APACHE-2.0	https://github.com/serde-rs/json
63	Rust Tracing	tracing	0.1.41	MIT	https://github.com/tokio-rs/tracing
64	Rust Tracing	tracing-subscriber	0.3.19	MIT	https://github.com/tokio-rs/tracing
65	Rust System	procfs	0.17.0	MIT OR APACHE-2.0	https://github.com/eminence/procfs
66	Android Visualization	com.google.accompanist:accompanist-drawablepainter	0.15.0	Apache 2.0	https://github.com/google/accompanist/tree/main/drawablepainter
67	Rust Async	async-broadcast	0.7.1	MIT OR APACHE-2.0	https://github.com/smol-rs/async-broadcast
68	Rust TUI	console	0.15.8	MIT	https://github.com/console-rs/console
69	Rust TUI	dialoguer	0.11.0	MIT	https://github.com/console-rs/dialoguer
70	Rust TUI	indicatif	0.17.9	MIT	https://github.com/console-rs/indicatif
71	Rust Raw Linux APIS	nix	0.29.0	MIT	https://github.com/nix-rust/nix
72	Gradle Plugin	com.android.tools.build:gradle	8.7.2	APACHE-2.0	https://maven.google.com/web/index.html?q=com.android.tools.build#com.android.tools.build:gradle:8.7.2
73	Android Coroutines	org.jetbrains.kotlinx:kotlinx-coroutines-android	1.9.0	APACHE-2.0	https://github.com/Kotlin/kotlinx.coroutines
74	Rust Safety	bytemuck	1.20.0	MIT OR APACHE-2.0 OR Zlib	https://github.com/Lokathor/bytemuck
75	Rust Concurrency	ractor	0.13.4	MIT	https://github.com/slawlor/ractor
76	Rust Concurrency	crossbeam	0.8.4	MIT OR APACHE-2.0	https://github.com/crossbeam-rs/crossbeam
77	Rust Parsing	object	0.36.5	MIR OR APACHE-2.0	https://github.com/gimli-rs/object
78	Static Analysis	detekt	1.23.7	APACHE-2.0	https://github.com/detekt/detekt

#	Context	Name	Version	License	Comment
79	Static Analysis	detekt-rules	0.0.26	APACHE-2.0	https://github.com/detekt/detekt
80	Business Logic	flowredux	1.2.2	APACHE-2.0	https://github.com/freeletics/FlowRedux
81	Business Logic	arrow	1.2.4	APACHE-2.0	https://github.com/arrow-kt/arrow
82	Rust Ebpf	aya-obj	0.2.1	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
83	Rust Ebpf	aya-log-common	0.1.15	MIT OR APACHE-2.0	https://github.com/aya-rs/aya
84	Rust Async	async-walkdir	2.0.0	APACHE-2.0	https://github.com/ririsoft/async-walkdir
85	Rust Async	tokio-process-stream	0.4.0	MIT	https://github.com/lpenz/tokio-process-stream
86	Symbols	symbolic	12.12.3	MIT	https://github.com/getsentry/symbolic
87	Symbols	tantivy	0.22.0	MIT	https://github.com/quickwit-oss/tantivy
88	Rust Async	fmmap	0.3.3	APACHE-2.0	https://github.com/al8n/fmmap
89	Rust Async	flume	0.11.1	MIT OR APACHE-2.0	https://github.com/zesterer/flume
90	Rust Android	adb-client	2.1.0	MIT	https://github.com/cocool97/adb_client
91	Rust Utils	ctrlc	3.4.5	MIT OR APACHE-2.0	https://github.com/Detegr/rust-ctrlc
92	Rust Linux	rustix	0.38.43	Apache-2.0 WITH LLVM-exception OR Apache-2.0 OR MIT	https://github.com/bytecodealliance/rustix
93	Rust Async	tower	0.5.2	MIT	https://github.com/tower-rs/tower
94	Rust Async	hyper-util	0.1.10	MIT	https://github.com/hyperium/hyper-util
95	Symbols	clang	2.0.0	APACHE-2.0	https://github.com/KyleMayes/clang-rs
96	Symbols	clang-sys	1.8.1	APACHE-2.0	https://github.com/KyleMayes/clang-sys

Last Name	First Name	Value					
Krug	Maximilian			2.00	#DIV/0!		
Ayach	Mohammed Tamim						
Bretting	Luca	2					
Seidl	Robin						
Hilgers	Felix	?		0	No size		
Weisshuhn	Tom			1	Trivial size		
Schlicht	Franz			2	Small size		
Nawlo	Ali			3	Medium size		
Zinn	Benedikt	?		5	Large size		
				8	Very large size		
				13	Too large (size)		
Team members left							
Labroussis	Christos						
How to play planning poker							
1. Everyone type their number into their value field, don't hit return yet							
2. Someone, perhaps a product owner, count down 3.. 2.. 1..							
3. Then, everyone hit return to submit their value							