Results of Document Analysis Using Thematic Analysis Approach[1]

Index	Initial codes	Higher order codes/categories	Emerging themes
1	I'd really prefer to work with somebody else "to more of a" look	overloaded maintainers	projects
2	sharing more work so that you can scale back sometimes	overloaded maintainers	projects
3	For a busy subsystem, can often be more than one person can handle.	overloaded maintainers	projects
4	By this time, Linux is no longer a hobbyist project, and after 21 years, it is probably time to focus more on scaling the maintainer role.	overloaded maintainers	projects
5	Maintainers are not keeping up with the kernel growth overall.	overloaded maintainers	projects
6	Most subsystems have unsustainable maintainer ratios.	overloaded maintainers	projects
7	cult of busy	overloaded maintainers	projects
8	Being a leader of a much bigger team makes maintainers very busy.	overloaded maintainers	projects
9	Community keeps growing, or you're your maintainer becomes otherwise busy with work&life.	overloaded maintainers	projects
10	Those maintainers lack reviewers.	overloaded maintainers	applicable projects
11	You have your standard-issue overloaded bottleneck.	overloaded maintainers	applicable projects
12	I'd argue that having a group would be substantially more robust.	single point of failure	projects
13	(A single maintainer) is hard to prepare for disaster.	single point of failure	projects
14	For a busy subsystem, can often be more than one person can handle.	single point of failure	projects
15	Group models are also more robust in the face of vacations, illness, or simply a day job that gets busy.	single point of failure	projects

16	He and Jani were becoming a bottleneck in the process.	single point of failure	projects
17	Most maintainers are just that, a single person, and often responsible for a bunch of different areas in the kernel with corresponding different git branches.	single point of failure	projects
18	The maintainer as bottleneck.	single point of failure	projects
19	bottleneck	single point of failure	projects
20	Has contributed at least 25 patches.	suitable candidates	committers
21	(Committers) should have submitted non-trivial patches.	suitable candidates	committers
22	(The patches should) being merged	suitable candidates	committers
23	(Committers) should have reviewed at least 25 patches.	suitable candidates	committers
24	Committers have enough experience.	suitable candidates	committers
25	A subsystem clearly needs a team of developers, and non-maintainer reviews must be the norm.	suitable candidates	committers
26	(Committers) should not abuse of commit rights.	trust among maintainers and committers	committers
27	Maintainers trust contributors.	trust among maintainers and committers	committers
28	(Committers) should be regular contributors.	trust among maintainers and committers	committers
29	Maintainers trust each other.	trust among maintainers and committers	committers
30	Trust is obviously key within the group, no matter background/ employment/ representation.	trust among maintainers and committers	committers
31	Trust relationships have to be built first.	trust among maintainers and committers	committers

32	People you would trust enough to do it.	trust among maintainers and committers	committers
33	particular if due to lack of trust	trust among maintainers and committers	committers
34	He (maintainer) trusts his committers.	trust among maintainers and committers	committers
35	It is a "human nature thing."	trust among maintainers and committers	committers
36	The group should be consistent, with developers who stay around.	trust among maintainers and committers	committers
37	Hardware for testing	sufficient precommit testing	risk mitigation
38	We have clearly documented merge criteria.	sufficient precommit testing	risk mitigation
39	We have massive CI, available to all contributors automatically.	sufficient precommit testing	risk mitigation
40	mandatory in-depth testing way before committing	sufficient precommit testing	risk mitigation
41	Good testing is crucial to this model.	sufficient precommit testing	risk mitigation
42	A multi-committer tree can never be rebased, so there is no way to remove embarrassing mistakes.	sufficient precommit testing	risk mitigation
43	Testing Requirements for drm/i915 Features and Patches	sufficient precommit testing	risk mitigation
44	Have confidence in the patches you push.	strict review process risk	risk mitigation
45	The confidence must be explicitly documented with special tags (Reviewed-by, Acked-by, Tested-by, Bugzilla, etc.) in the commit message.	strict review process risk	risk mitigation
46	The complexity and impact are properties of the patch that must be justified in the commit message.	strict review process risk	risk mitigation

47	One of those is mandatory review, no one is allowed to do anything solo.	strict review process risk	risk mitigation
48	Especially around purported quality enforcement tools like code reviews.	strict review process risk	risk mitigation
49	dim: drm inglorious maintainer script	application of tools to simplify work and reduce errors	risk mitigation
50	advanced commands for committers and maintainers	application of tools to simplify work and reduce errors	risk mitigation
51	Pipes stdin into the fixup patch file for the current drm-Ip merge. A branch can be explicitly specified to fix up a non-conflicting tree that fails to build.	application of tools to simplify work and reduce errors	risk mitigation
52	This command adds the Link: tag for patches that failed to apply directly).	application of tools to simplify work and reduce errors	risk mitigation
53	Any duplicates by name or email will be removed automatically.	application of tools to simplify work and reduce errors	risk mitigation
54	Using patchwork to facilitate review.	application of tools to simplify work and reduce errors	risk mitigation
55	quilt git flow script facilitates review.	application of tools to simplify work and reduce errors	risk mitigation
56	qf is a workflow script to manage a quilt patch pile on top of a git baseline and track any changes in git itself.	application of tools to simplify work and reduce errors	risk mitigation
57	This automaEcally either creates a new, empty patch pile or checks out the state of an exisEng remote.	application of tools to simplify work and reduce errors	risk mitigation
58	tooling	application of tools to simplify work and reduce errors	risk mitigation
59	purported quality enforcement tools	application of tools to simplify work and reduce errors	risk mitigation

60	tools are necessary	application of tools to simplify work and reduce errors	risk mitigation
61	When somebody makes a mistake, if possible, a check should be put into the tools to keep it from happening again.	application of tools to simplify work and reduce errors	risk mitigation

1. Soares C D , Dybå Tore. Recommended Steps for Thematic Synthesis in Software Engineering[C]// International Symposium on Empirical Software Engineering & Measurement. IEEE, 2011.