Results of Document Analysis Using Thematic Analysis Approach[1]

Initial codes	Higher order codes/categories	Emerging themes
I'd really prefer to work with somebody else "to more of a" look	overloaded maintainers	projects
sharing more work so that you can scale back sometimes	overloaded maintainers	projects
For a busy subsystem, can often be more than one person can handle.	overloaded maintainers	projects
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
Maintainers are not keeping up with the kernel growth overall.	overloaded maintainers	projects
Most subsystems have unsustainable maintainer ratios.	overloaded maintainers	projects
cult of busy	overloaded maintainers	projects
Being a leader of a much bigger team makes maintainers very busy.	overloaded maintainers	projects
Community keeps growing, or your maintainer becomes otherwise busy with work&life.	overloaded maintainers	projects
Those maintainers lack reviewers.	overloaded maintainers	projects
You have your standard-issue overloaded bottleneck.	overloaded maintainers	projects
I'd argue that having a group would be substantially more robust.	single point of failure	projects
(A single maintainer) is hard to prepare for disaster.	single point of failure	projects
For a busy subsystem, can often be more than one person can handle.	single point of failure	projects
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
He and Jani were becoming a bottleneck in the process.	single point of failure	projects

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
You have your standard-issue overloaded bottleneck.	single point of failure	projects
The maintainer as bottleneck.	single point of failure	projects
bottleneck	single point of failure	projects
Has contributed at least 25 patches.	suitable candidates	committers
(Committers) should have submitted non-trivial patches.	suitable candidates	committers
(The patches should) being merged	suitable candidates	committers
(Committers) should have reviewed at least 25 patches.	suitable candidates	committers
Committers have enough experience.	suitable candidates	committers
A subsystem clearly needs a team of developers, and non-maintainer reviews must be the norm.	suitable candidates	committers
(Committers) should not abuse of commit rights.	trust among maintainers and committers	committers
Maintainers trust contributors.	trust among maintainers and committers	committers
(Committers) should be regular contributors.	trust among maintainers and committers	committers
Maintainers trust each other.	trust among maintainers and committers	committers
Trust is obviously key within the group, no matter background/ employment/ representation.	trust among maintainers and committers	committers
Trust relationships have to be built first.	trust among maintainers and committers	committers
People you would trust enough to do it.	trust among maintainers and committers	committers

particular if due to lack of trust	trust among maintainers and committers	committers
He (maintainer) trusts his committers.	trust among maintainers and committers	committers
It is a "human nature thing."	trust among maintainers and committers	committers
The group should be consistent, with developers who stay around.	trust among maintainers and committers	committers
Hardware for testing	sufficient precommit testing	risk mitigation
We have clearly documented merge criteria.	sufficient precommit testing	risk mitigation
We have massive CI, available to all contributors automatically.	sufficient precommit testing	risk mitigation
mandatory in-depth testing way before committing	sufficient precommit testing	risk mitigation
Good testing is crucial to this model.	sufficient precommit testing	risk mitigation
A multi-committer tree can never be rebased, so there is no way to remove embarrassing mistakes.	sufficient precommit testing	risk mitigation
Testing Requirements for drm/i915 Features and Patches	sufficient precommit testing	risk mitigation
Have confidence in the patches you push.	strict review process	risk mitigation
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 mitigation
The complexity and impact are properties of the patch that must be justified in the commit message.	strict review process	risk mitigation
One of those is mandatory review, no one is allowed to do anything solo.	strict review process	risk mitigation
Especially around purported quality enforcement tools like code reviews.	strict review process	risk mitigation

dim: drm inglorious maintainer script	application of tools to simplify work and reduce errors	risk mitigation
advanced commands for committers and maintainers	application of tools to simplify work and reduce errors	risk mitigation
Pipes stdin into the fixup patch file for the current drm-Ip merge. A branch can be explicitly specified to fix up a non-conflicIng tree that fails to build.	application of tools to simplify work and reduce errors	risk mitigation
This command adds the Link: tag (for patches that failed to apply directly).	application of tools to simplify work and reduce errors	risk mitigation
Any duplicates by name or email will be removed automatically.	application of tools to simplify work and reduce errors	risk mitigation
Using patchwork to facilitate review.	application of tools to simplify work and reduce errors	risk mitigation
quilt git flow script facilitates review.	application of tools to simplify work and reduce errors	risk mitigation
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
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
tooling	application of tools to simplify work and reduce errors	risk mitigation
purported quality enforcement tools	application of tools to simplify work and reduce errors	risk mitigation
tools are necessary	application of tools to simplify work and reduce errors	risk mitigation
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.