The Path to Achieve a Bug-Free Build on the Mainline

Philip Li philip.li@intel.com

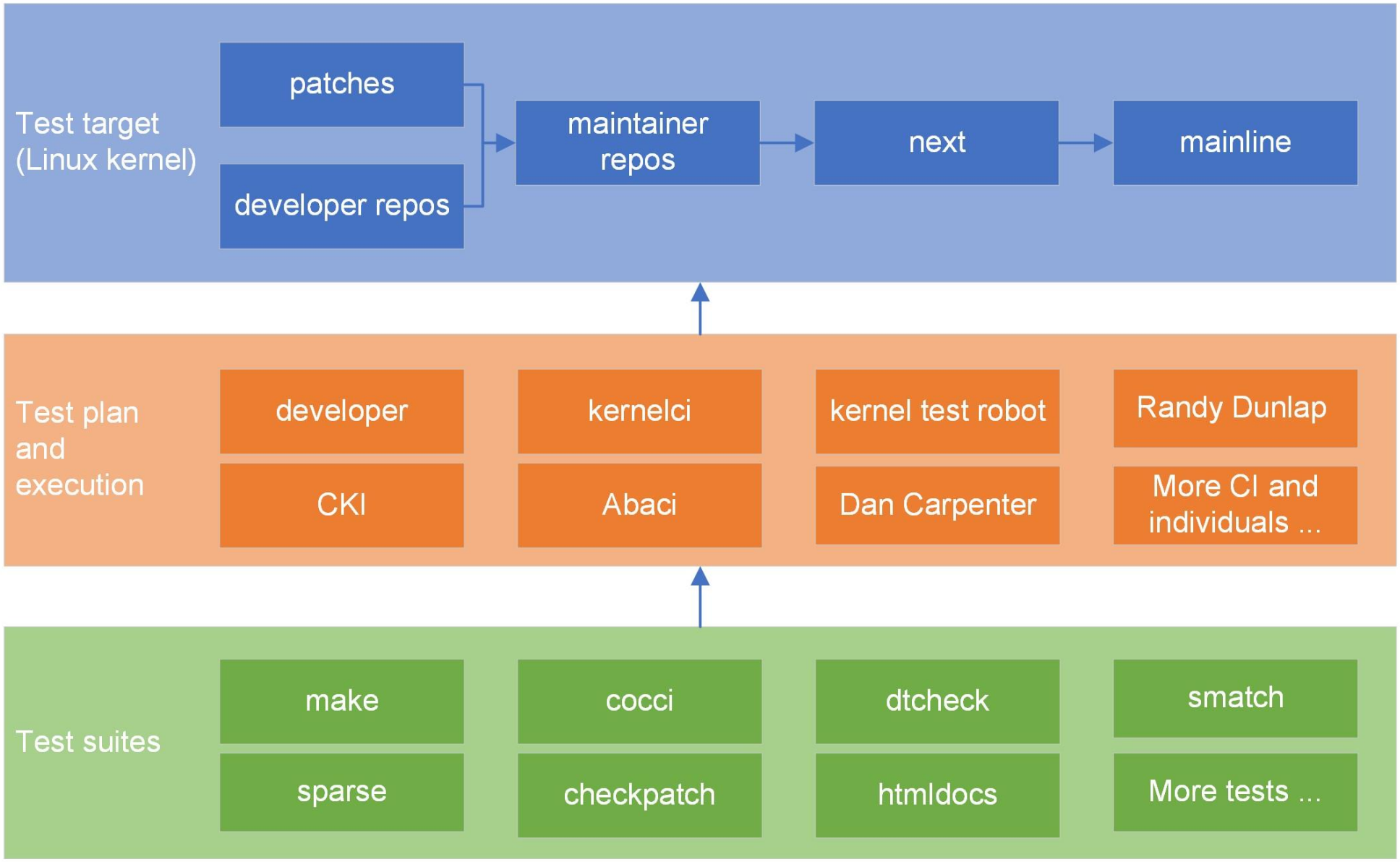
Agenda

- The status quo of build test
- The tactical practices in 0-Day CI
- The challenges and proposals

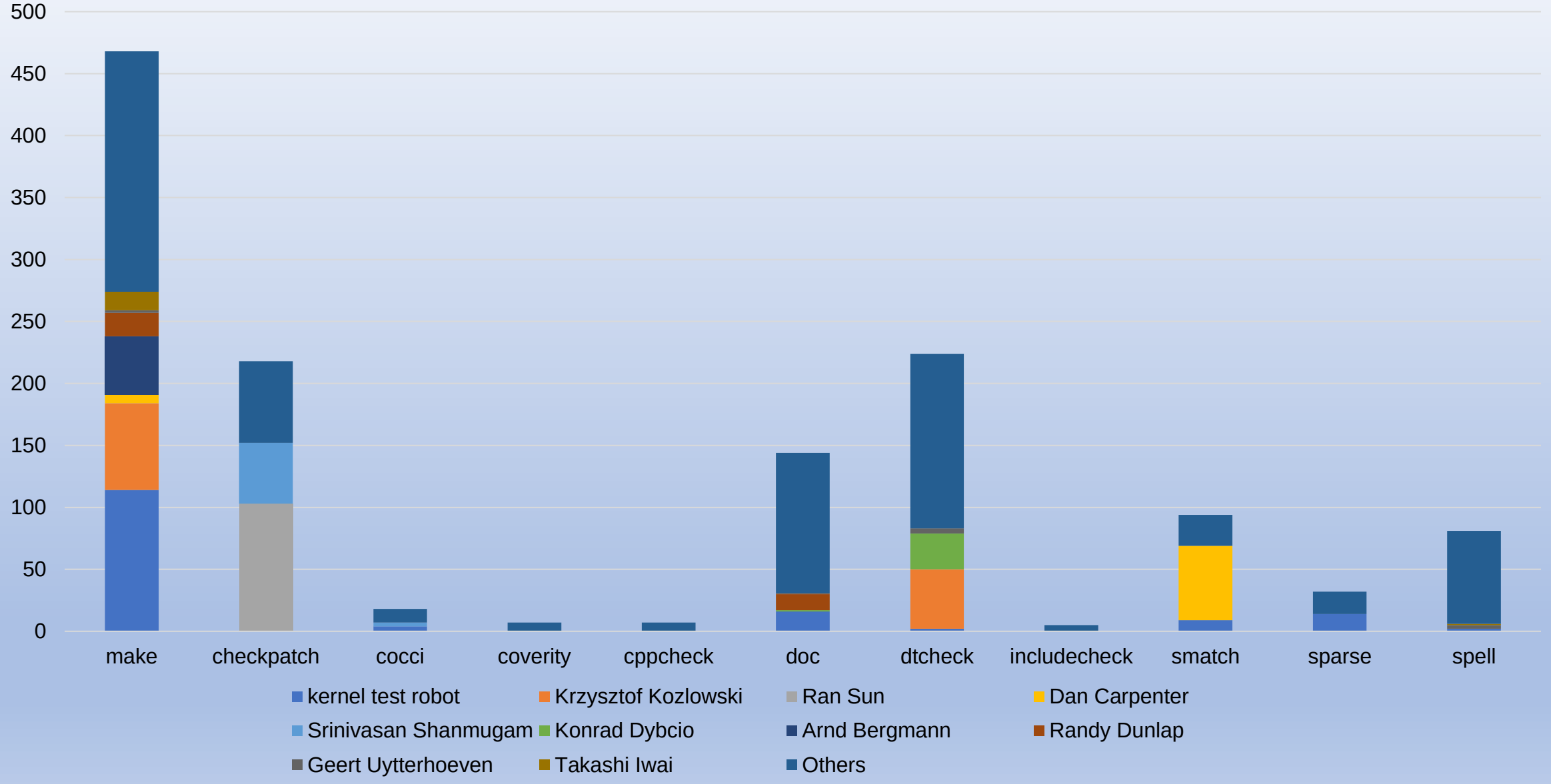
Agenda

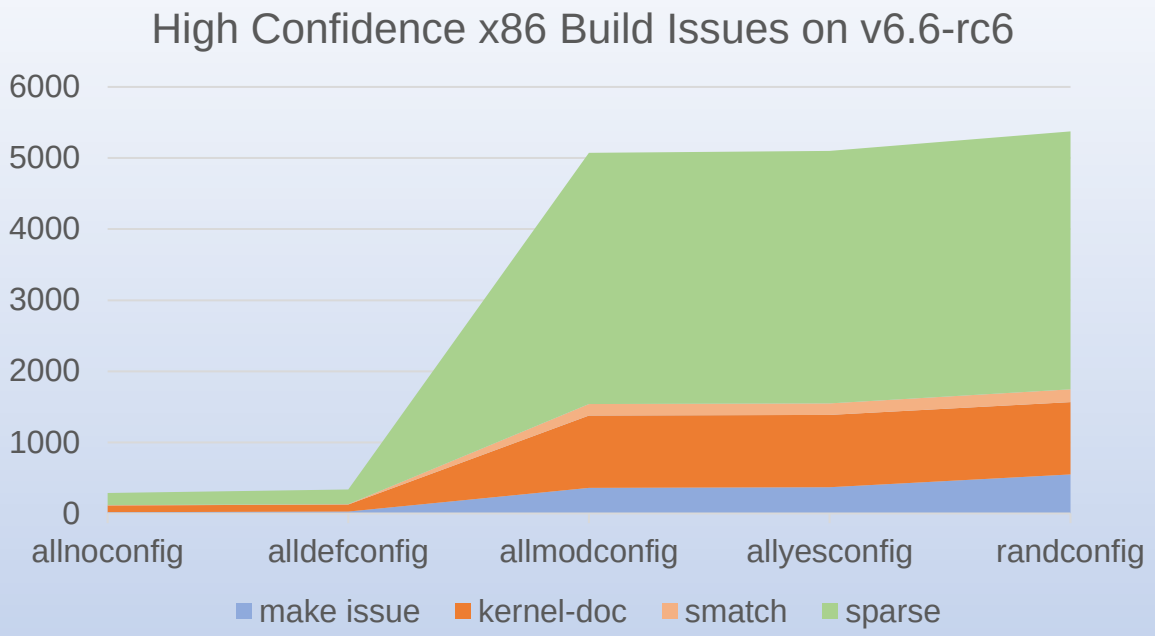
- The status quo of build test
- The tactical practices in 0-Day CI
- The challenges and proposals

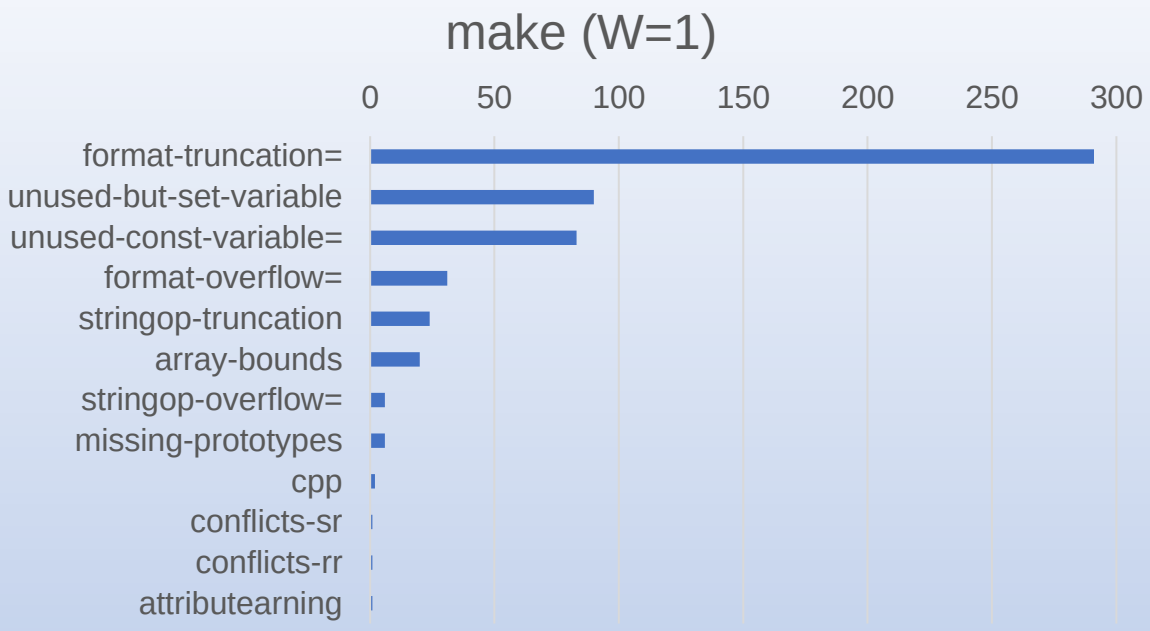
Build Test Effort

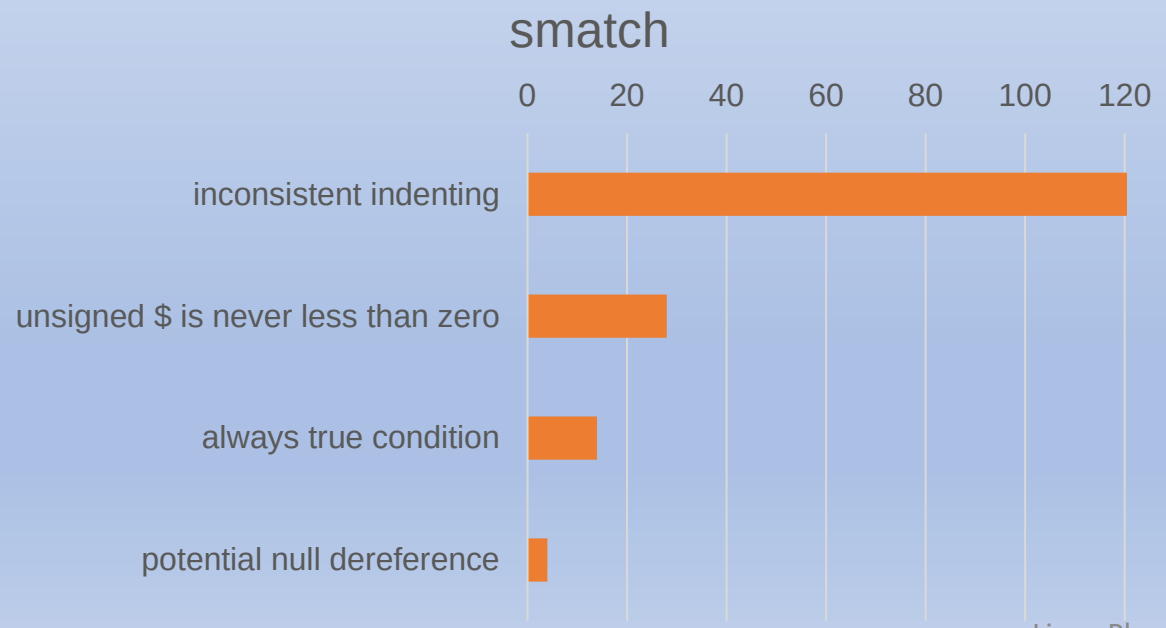


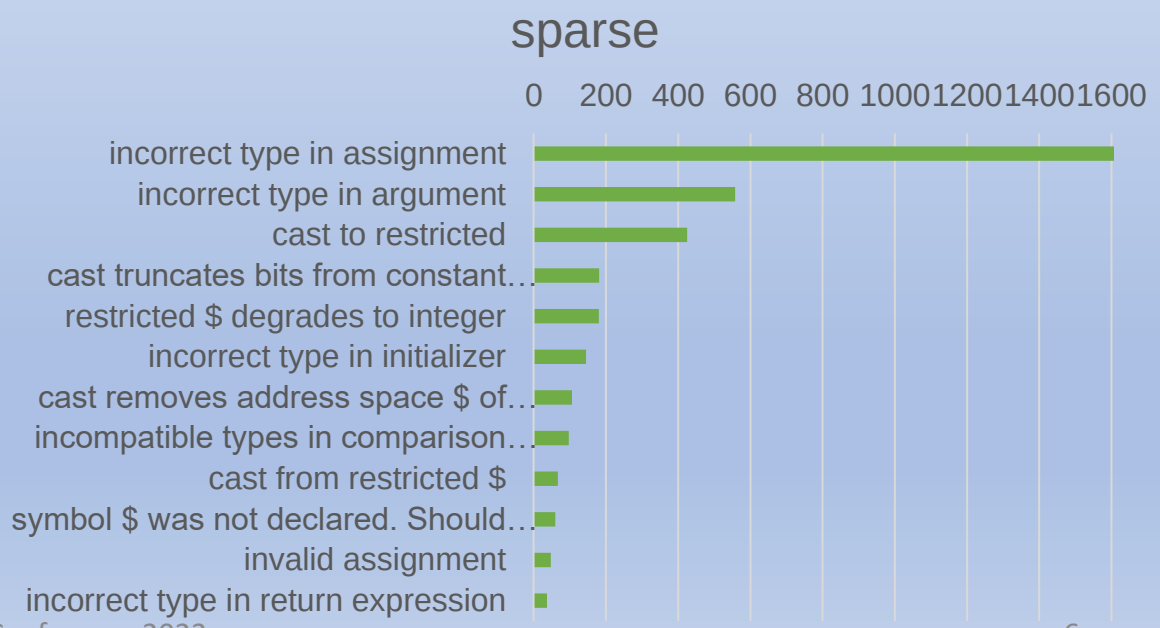
Tools Used to Detect Build Issues [v6.6-rc1 - v6.6-rc6]




High Confidence x86 Build Issues on v6.6-rc6
6000
5000
4000
3000
2000
1000
0
allnoconfig
alldefconfig
allmodconfig
allyesconfig
randconfig
make issue
kernel-doc
smatch
sparse


make (W=1)
0
50
100
150
200
250
300
format-truncation=
unused-but-set-variable
unused-const-variable=
format-overflow=
stringop-truncation
array-bounds
stringop-overflow=
missing-prototypes
cpp
conflicts-sr
conflicts-rr
attributearning


smatch
0
20
40
60
80
100
120
inconsistent indenting
unsigned $ is never less than zero
always true condition
potential null dereference

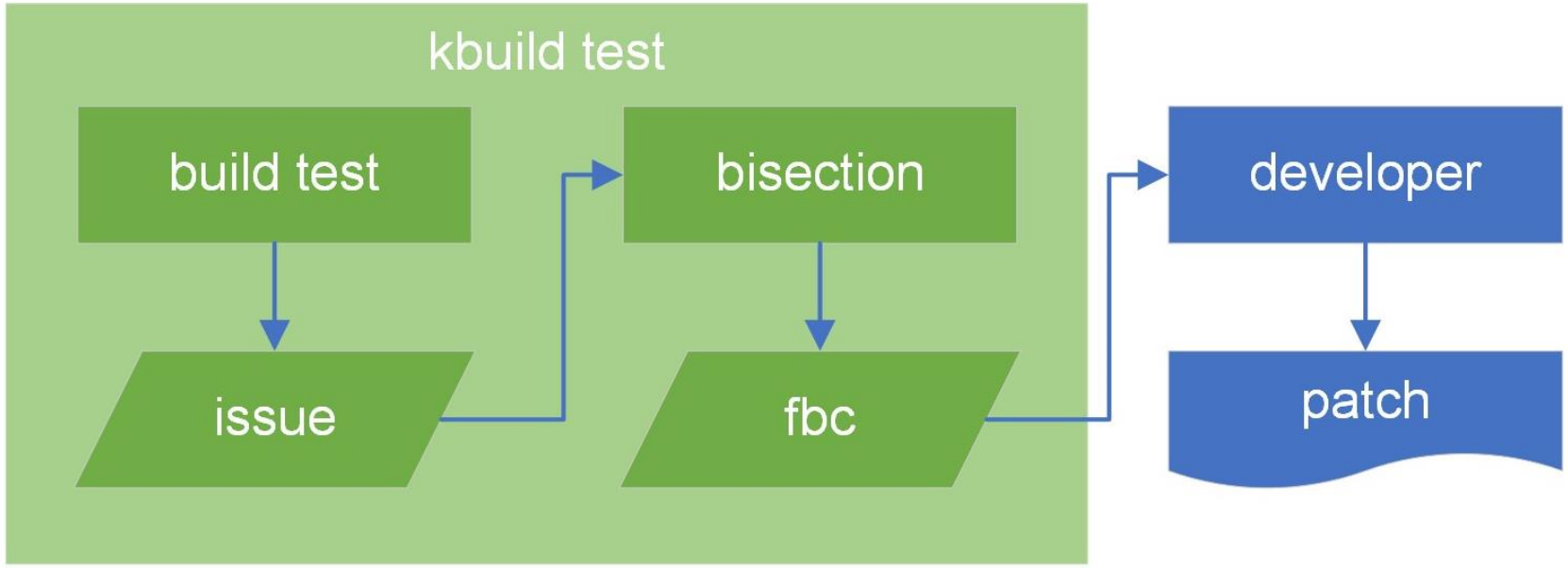

sparse
0
200
400
600
800
1000
1200
1400
1600
incorrect type in assignment
incorrect type in argument
cast to restricted
cast truncates bits from constant…
restricted $ degrades to integer
incorrect type in initializer
cast removes address space $ of…
incompatible types in comparison…
cast from restricted $
symbol $ was not declared. Should…
invalid assignment
incorrect type in return expression

Some Build Issues are Escaped to Mainline

- Patch in mailing list is not all build tested by CI.
- Issue is detected on immutable branch.
- Build coverage is difficult to reach 100%, like W=1, randconfig on different archs, static analysis tools.
- Some warning options are newly enabled, like -Wformat-truncation.
- COMPILE_TEST is not yet fully opened.

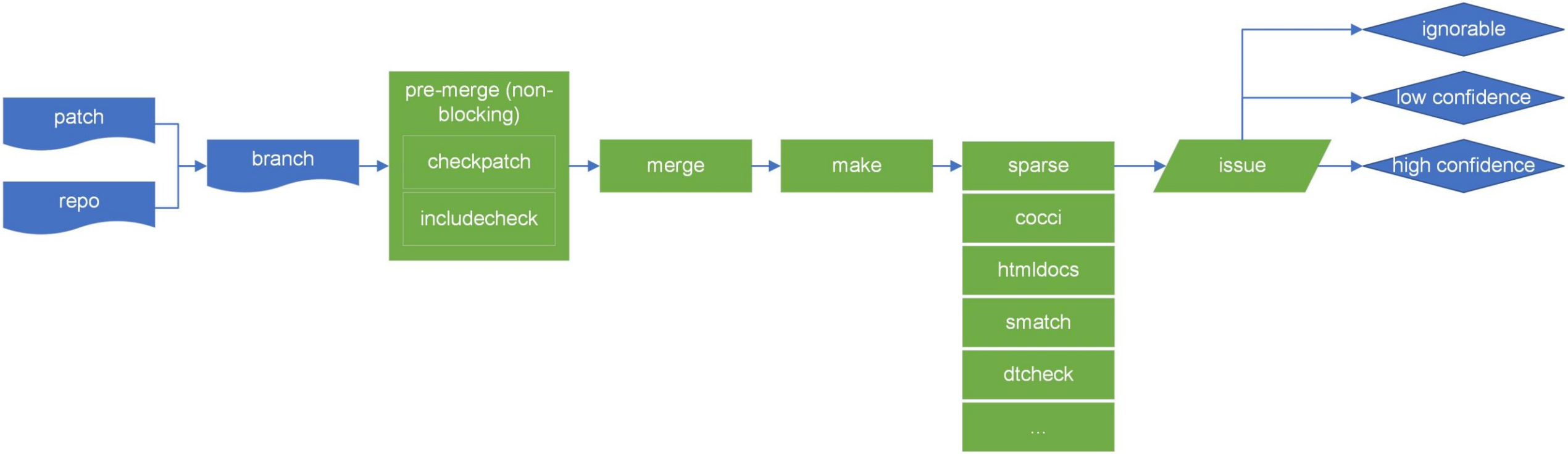
Agenda

- The status quo of build test
- The tactical practices in 0-Day CI
- The challenges and proposals


kbuild test
build test
bisection
developer
issue
fbc
patch

Build Test

Shift Left (More repos + More test suites)



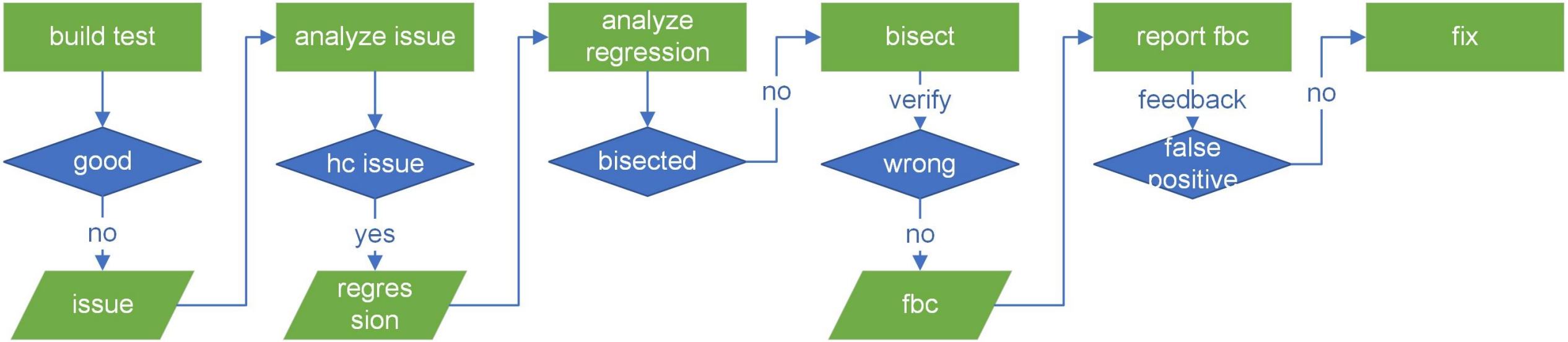
The Best Effort to Reduce False Positive

- Patterns for different confidence levels, which are gathered from
  - Developer feedback
  - The changelog of fixed issue on mainline
  - The input from the maintainer of static analysis tool

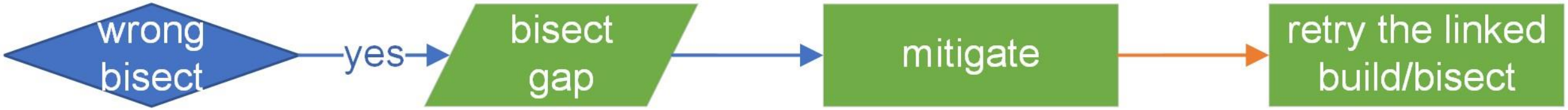
```
checkpatch low confidence patterns

added, moved or deleted file\(s\), does MAINTAINERS need updating\?
adding a line without newline at end of file
__func__ should be used instead of gcc specific __FUNCTION__
line over 80 characters
LINUX_VERSION_CODE should be avoided, code should be for the version
Missing commit description - Add an appropriate one
please write a help paragraph that fully describes the config symbol
Possible repeated word: 'Google'
Possible unwrapped commit description \(prefer a maximum 75 chars per line\)
please, no spaces at the start of a line
```

Bisection

Assist developer to ease the analysis



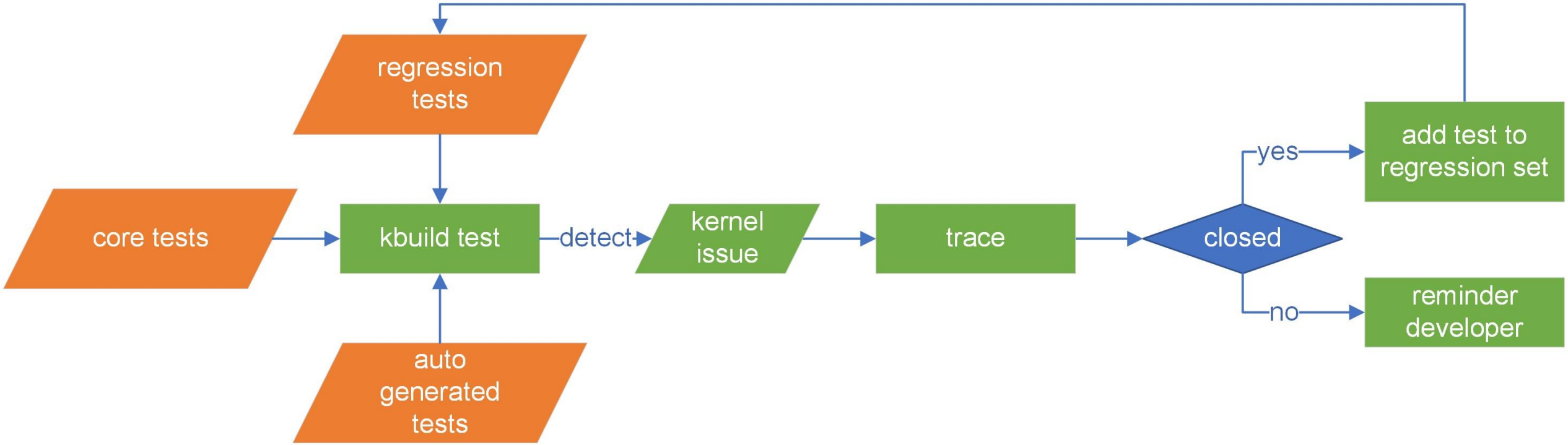
Bisection Tuning



Build Bisectability	Examples	Mitigation
Makefile	[v5.8-rc7] 20b1be595282 ("kbuild: fix single target builds for external modules")	
	[v6.6-rc1] 6d4ab2e97dcf ("extrawarn: enable format and stringop overflow warnings in W=1")	Specify option in KCFLAGS
Build error	[v5.6] devicetree-fixes-for-5.6-4 ("scripts/dtc: Remove redundant YYLOC global declaration")	Patch the script
Compiler option change	[v5.18-rc1] cf300b83c793 ("kbuild: Remove '-mno-global-merge'")	Patch the Makefile
Compiler new feature	[v5.16-rc3] 310780e825f3 ("usb: dwc2: hcd_queue: Fix use of floating point literal")	
Existing error pattern change	[v6.6-rc1] 15fa3e8e3269 ("mips: implement the new page table range API")	
Existing error exposed	[v6.0-rc1] 600b7b26c07a ("tools bpftool: Fix compilation error with new binutils")	
Scripts new feature	[v5.14-rc1] 7845daa8bd72 ("coccinelle: misc: add swap script")	Use latest version
	[v6.2-rc1] 03699f271de1 ("string: Rewrite and add more kern-doc for the str*() functions")	Stick to bad commit's version

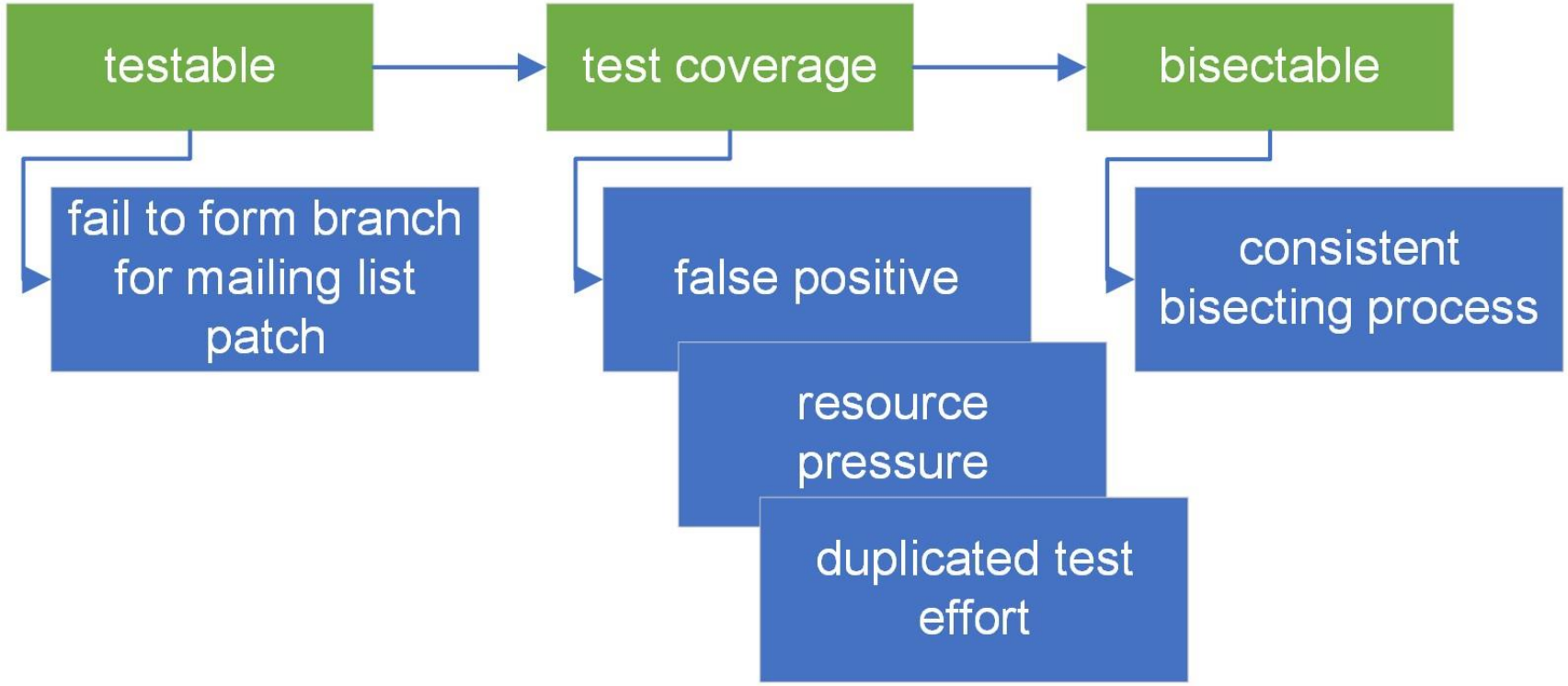
Build Test Tuning

Drive the closure and avoid regression



Agenda

- The status quo of build test
- The tactical practices in 0-Day CI
- The challenges and proposals


testable
test coverage
bisectable
fail to form branch for mailing list patch
false positive
resource pressure
duplicated test effort
consistent bisecting process

Short Term	Proposal	Remark
Testable	The patch series supplies the base in a standardized way.	• e.g. git-format-patch --base, or specify base tag in cover letter like Based-on-link • https://github.com/intel-lab-lkp/linux/branches • https://patchwork.ozlabs.org/
	A public repo that contains the commits	
Test coverage	Execute make with W=1	
	Shift left	• Leverage merging approach like linux-next • https://github.com/intel/lkp-tests/tree/master/repo/linux
	randconfig	
	Static analysis tools	Maintain the patterns for different confidence levels

Long Term	Proposal	Remark
Testable	Convention to communicate with bot	• Named branches: -for-ci, -for-next, -urgent, -no-test, etc • Repo configuration: https://github.com/intel/lkp-tests/wiki/Repo-Spec • https://github.com/google/syzkaller/blob/master/docs/syzbot.md#communication-with-syzbot
	Open more COMPILE_TEST	
Test coverage	Different versions of compilers	
	More static analysis tools	• Keep false positive in minimum level
	AI guided for test effectiveness	• Test selection per code change, https://engineering.fb.com/2018/11/21/developer-tools/predictive-test-selection/
	Smoke4Dev test kit	• Avoid the known blocking issues • Run “locally” in a reasonable time, by git hook or github action
Bisectable	Tool for consistent build bisection	

Discussion

Thanks