Test reporting tool (TRT)

Motivations, goals and requirements

What I did

4 semi-structured interviews in early August 2016:

- 2 with the Intel QA team
- 2 with the Intel development team

What I found

Although there are many commonalities between the motivations, goals and information requirements of the QA and development teams, there is a fundamental difference in scope.

While QA is primarily concerned with the presentation and manipulation of data from release candidate builds, development is primarily concerned with the presentation and manipulation of data from any build run by the Autobuilder (not just release candidate builds). It is continuous integration data they are after.

In addition to test results and performance metrics, developers are also interested in build output data currently collected by build history (images, their size, file and package content).

What does this mean?

Satisfying the requirements of the development team adds significant complexity:

1. It means storing and manipulating much more data, which is likely to bring up performance problems.

2. It requires integrating pretty much all our data presentation tools, including buildhistory-web, the error reporting tool and Toaster.

From a design perspective at least, this is a much bigger problem to tackle.

What do we do?

Focus on the goals and requirements of the QA team first, which are better defined and more limited in scope.

Since they are a subset of the developers' goals and requirements, everybody wins.

Motivations and goals

- Improve planning and resource allocation.
- Increase focus on the most problematic project components and test cases.
- Reduce the amount of manual work.
- Faster delivery of reports and QA information to the project maintainers.
- Improve the quality of the information delivered to the project maintainers, pointing to problematic areas and including quantitative data.
- Standardise output and presentation of results for all test runners.
- Keep a historical record of QA data (store all test results for all release candidates).
- Overcome the presentation constraints and data analysis limitations imposed by the existing tooling (Bugzilla, Testopia and MediaWiki).

Information requirements

- See test results for a single release candidate.
- See test results for several release candidates.
- See the overall status of one or more releases candidates at a glance.
- For each test result, see associated test cases, test runs, Bugzilla issues and component information.
- Search test results and its associated information using free text queries.

For one or more release candidates:

- See test results for a subset of test outcomes (passed, failed, blocked, skipped).
- See test results for a subset of components.
- See test results for a component's subset of environments.
- See a subset of test results for any combination of the above 3 criteria (outcome + component + environment).

Data sources and tools

- Testopia (test runs, test cases and associated Bugzilla issues).
- Bugzilla (issue descriptions and importance).
- Build performance data.
- Test runners (xml output).
- Autobuilder (build logs, release candidate images).
- Error reporting tool (error reports logged for a release candidate).

Motivations and goals

- Enable analysis of data we already collect but is currently scattered around different tools and Git repositories.
- Visualise evolution of project quality over time.
- Better problem detection. The goal is to proactively identify issues, instead of relying on community reports.
- Aid release decision making, by helping answer the question: is this candidate fit for release?
- Better planning and more effective distribution of QA effort.
- Reduce manual work in the QA workflow.
- Improve the information QA feeds to the project maintainers.
- Increase visibility of project status within the community.
- Increase visibility and appreciation of existing QA effort within by the community.

Information requirements

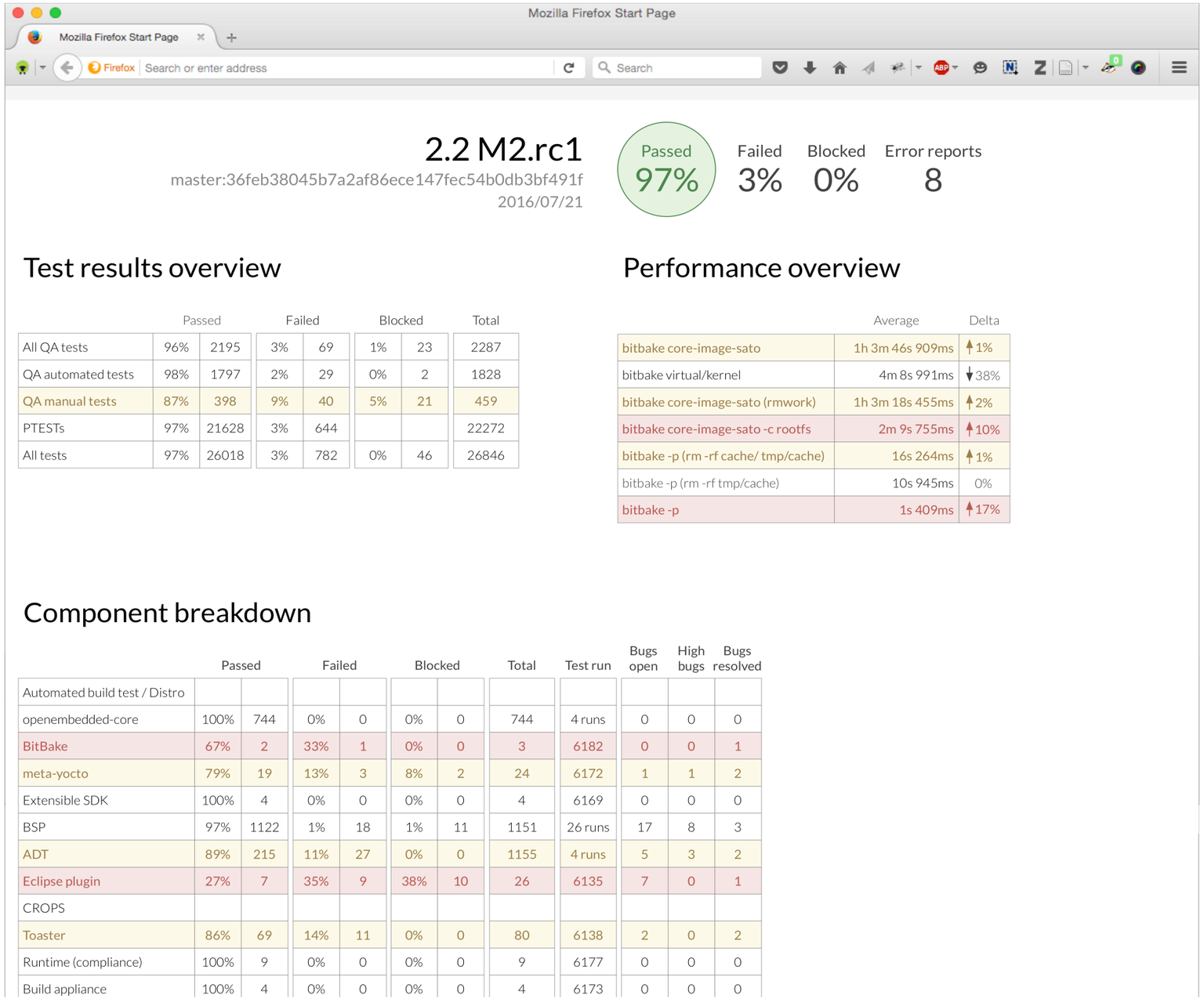
All the ones listed for the QA team, plus the following additional requirements:

- See build output data provided by build history, not just tests results (e.g. image size, number of files in rootfs, file list and location, package dependencies, etc).
- See build output data, performance data and test results for all builds executed in the Autobuilder, not just for release candidates.
- See the above information not just for one build, but for several builds, for comparison purposes.
- For any build executed in the Autobuilder, see all the error reports generated.

Data sources and tools

All the ones listed for the QA team, plus:

- Build history for image size, rootfs content and package dependencies.
- buildhistory-web, for visualising differences between builds.
- Toaster, which already displays image size, rootfs content and package dependencies.



2.2 M2.rc1

master:36feb38045b7a2af86ece147fec54b0db3bf491f
2016/07/21

Passed	Failed	Blocked	Error reports
97%	3%	0%	8

Test results overview

	Passed		Failed		Blocked		Total
All QA tests	96%	2195	3%	69	1%	23	2287
QA automated tests	98%	1797	2%	29	0%	2	1828
QA manual tests	87%	398	9%	40	5%	21	459
PTESTs	97%	21628	3%	644			22272
All tests	97%	26018	3%	782	0%	46	26846

Performance overview

	Average	Delta
bitbake core-image-sato	1h 3m 46s 909ms	↑1%
bitbake virtual/kernel	4m 8s 991ms	↓38%
bitbake core-image-sato (rmwork)	1h 3m 18s 455ms	↑2%
bitbake core-image-sato -c rootfs	2m 9s 755ms	↑10%
bitbake -p (rm -rf cache/ tmp/cache)	16s 264ms	↑1%
bitbake -p (rm -rf tmp/cache)	10s 945ms	0%
bitbake -p	1s 409ms	↑17%

Component breakdown

	Passed		Failed		Blocked		Total	Test run	Bugs open	High bugs	Bugs resolved
Automated build test / Distro											
openembedded-core	100%	744	0%	0	0%	0	744	4 runs	0	0	0
BitBake	67%	2	33%	1	0%	0	3	6182	0	0	1
meta-yocto	79%	19	13%	3	8%	2	24	6172	1	1	2
Extensible SDK	100%	4	0%	0	0%	0	4	6169	0	0	0
BSP	97%	1122	1%	18	1%	11	1151	26 runs	17	8	3
ADT	89%	215	11%	27	0%	0	1155	4 runs	5	3	2
Eclipse plugin	27%	7	35%	9	38%	10	26	6135	7	0	1
CROPS											
Toaster	86%	69	14%	11	0%	0	80	6138	2	0	2
Runtime (compliance)	100%	9	0%	0	0%	0	9	6177	0	0	0
Build appliance	100%	4	0%	0	0%	0	4	6173	0	0	0

Dashboard design proposal

The Test reporting tool UI will likely have the following main features:

- A way to select a specific release candidate

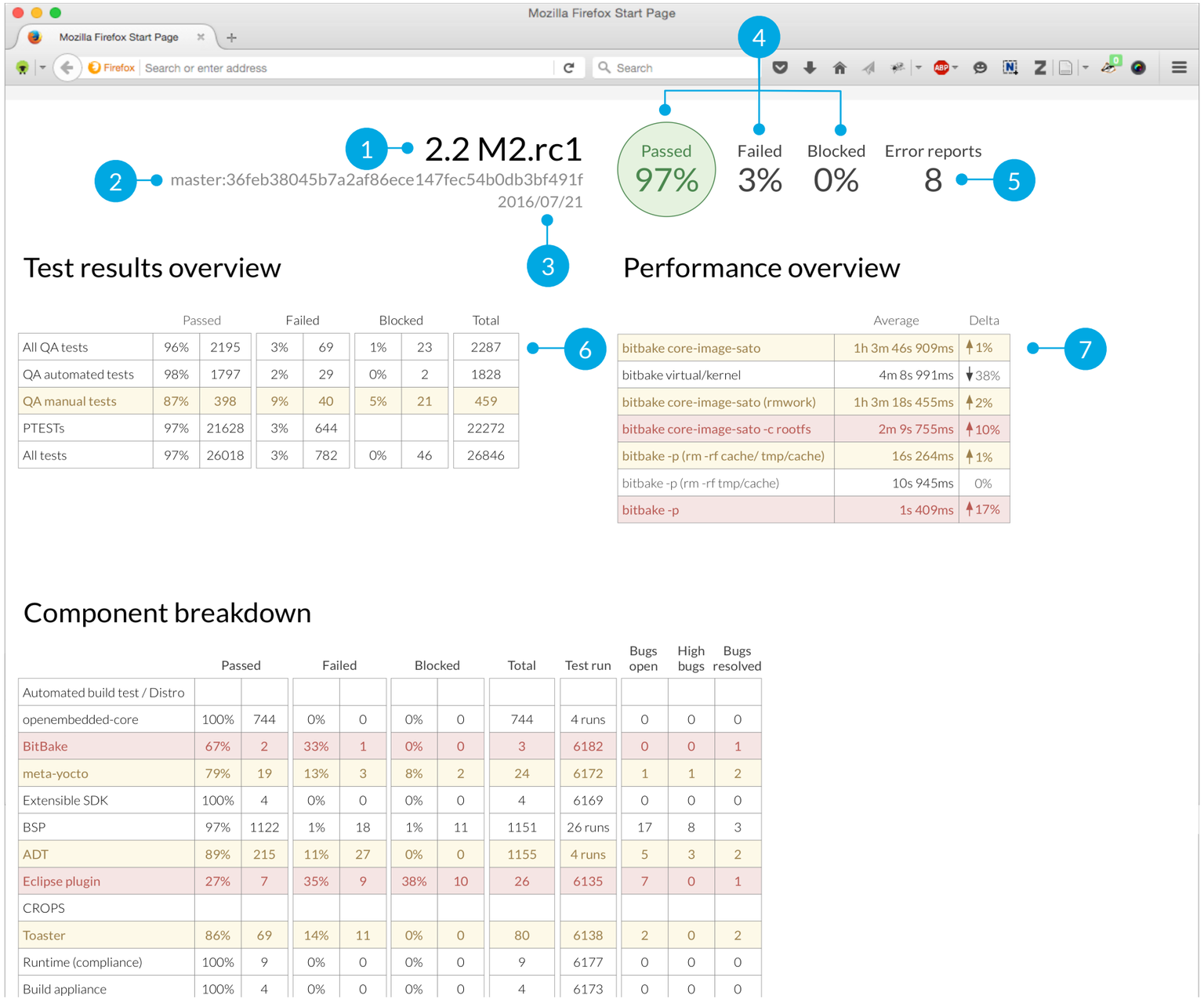
For each release candidate:

- A dashboard to provide an overview of the test results
- A table of test cases with good filtering capabilities, so that you can, for example, see all test cases for meta-yocto with medium+ or high bugs associated to them
- A separate table to show ptest results
- An additional page to display build performance information
- A comparison function that will allow you to see data for several release candidates

The image on the left shows a first design proposal for the release candidate dashboard. A higher resolution image is available at

https://wiki.yoctoproject.org/wiki/File:Dashboard.png

Dashboard content (I)



	Passed		Failed		Blocked		Total
All QA tests	96%	2195	3%	69	1%	23	2287
QA automated tests	98%	1797	2%	29	0%	2	1828
QA manual tests	87%	398	9%	40	5%	21	459
PTESTs	97%	21628	3%	644			22272
All tests	97%	26018	3%	782	0%	46	26846

	Average	Delta
bitbake core-image-sato	1h 3m 46s 909ms	↑1%
bitbake virtual/kernel	4m 8s 991ms	↓38%
bitbake core-image-sato (rmwork)	1h 3m 18s 455ms	↑2%
bitbake core-image-sato -c rootfs	2m 9s 755ms	↑10%
bitbake -p (rm -rf cache/ tmp/cache)	16s 264ms	↑1%
bitbake -p (rm -rf tmp/cache)	10s 945ms	0%
bitbake -p	1s 409ms	↑17%

Component breakdown

	Passed		Failed		Blocked		Total	Test run	Bugs open	High bugs	Bugs resolved
Automated build test / Distro											
openembedded-core	100%	744	0%	0	0%	0	744	4 runs	0	0	0
BitBake	67%	2	33%	1	0%	0	3	6182	0	0	1
meta-yocto	79%	19	13%	3	8%	2	24	6172	1	1	2
Extensible SDK	100%	4	0%	0	0%	0	4	6169	0	0	0
BSP	97%	1122	1%	18	1%	11	1151	26 runs	17	8	3
ADT	89%	215	11%	27	0%	0	1155	4 runs	5	3	2
Eclipse plugin	27%	7	35%	9	38%	10	26	6135	7	0	1
CROPS											
Toaster	86%	69	14%	11	0%	0	80	6138	2	0	2
Runtime (compliance)	100%	9	0%	0	0%	0	9	6177	0	0	0
Build appliance	100%	4	0%	0	0%	0	4	6173	0	0	0

Keep in mind this is only a first version, just a way to get us started. It is a static mock up, and does not include any navigation design.

The data shown in the image comes from the QA report for 2.2 M2.rc1. I hope using real data will help us make decisions about which content to include.

The dashboard shows:

1. Selected release candidate

2. Branch and commit

3. Date

4. Overall test results

5. Number of error reports in the error reporting tool for the tested commit SHA

6. Break down of tests results into automated, manual and ptests.

7. Performance data, including the difference over the previous release candidate

All tests	97%	20018	3%	782	0%	46	20846

bitbake -p (rm -rf cache/ tmp/cache)	16s 284ms	↑ 1%
bitbake -p (rm -rf tmp/cache)	10s 945ms	0%
bitbake -p	1s 409ms	↑ 17%

Component breakdown

	Passed		Failed		Blocked		Total	Test run	Bugs open	High bugs	Bugs resolved
Automated build test / Distro											
openembedded-core	100%	744	0%	0	0%	0	744	4 runs	0	0	0
BitBake	67%	2	33%	1	0%	0	3	6182	0	0	1
meta-yocto	79%	19	13%	3	8%	2	24	6172	1	1	2
Extensible SDK	100%	4	0%	0	0%	0	4	6169	0	0	0
BSP	97%	1122	1%	18	1%	11	1151	26 runs	17	8	3
ADT	89%	215	11%	27	0%	0	1155	4 runs	5	3	2
Eclipse plugin	27%	7	35%	9	38%	10	26	6135	7	0	1
CROPS											
Toaster	86%	69	14%	11	0%	0	80	6138	2	0	2
Runtime (compliance)	100%	9	0%	0	0%	0	9	6177	0	0	0
Build appliance	100%	4	0%	0	0%	0	4	6173	0	0	0

8

BSP breakdown

	Passed		Failed		Blocked		Total	Test run	Bugs open	High bugs	Bugs resolved
genericx86	96%	191	3%	7	0%	1	199	4 runs	7	3	0
genericx86_64	94%	293	3%	8	3%	10	311	6 runs	7	3	1
Beagle bone Black	100%	67	0%	0	0%	0	67	2 runs	0	0	1
EdgeRouter	100%	61	0%	0	0%	0	61	2 runs	0	0	0
MPC8315e-rdb	100%	62	0%	0	0%	0	62	2 runs	0	0	0
qemux86	97%	62	3%	2	0%	0	64	2 runs	1	1	1
qemux86_64	98%	63	2%	1	0%	0	64	2 runs	1	1	0
qemuarm	100%	127	0%	0	0%	0	127	2 runs	0	0	0
qemumips	100%	133	0%	0	0%	0	133	2 runs	1	0	0
qemuppc	100%	63	0%	0	0%	0	63	2 runs	0	0	0

9

8. Test results broken down per component

9. BSP test results broken down by BSP