Coccinelle: 10 Years of Automated Evolution in the Linux Kernel

Julia Lawall, Gilles Muller (Inria/LIP6)
October 23, 2018
The Linux kernel:

- Open source OS kernel, used in smartphones to supercomputers.
- 16MLOC and rapidly growing.
- Frequent changes to improve correctness and performance.

Issues:

- How to perform evolutions in such a large code base?
- Once a bug is found, how to check whether it occurs elsewhere?
How to better maintain large code bases?

**Patches:** The key to reasoning about change in the Linux kernel.

```c
@@ -1348,8 +1348,7 @@
- fh = kmalloc(sizeof(struct zoran_fh), GFP_KERNEL);
+ fh = kzalloc(sizeof(struct zoran_fh), GFP_KERNEL);
 if (!fh) {
   dprintk(1, KERN_ERR "%s: zoran_open(): allocation of zoran_fh failed\n",
          ZR_DEVNAME(zr));
 return -ENOMEM;
 }
- memset(fh, 0, sizeof(struct zoran_fh));
```
A SmPL idea: Raise the level of abstraction to semantic patches.

From:

@@ -1348,8 +1348,7 @@
- fh = kmalloc(sizeof(struct zoran_fh), GFP_KERNEL);
+ fh = kzalloc(sizeof(struct zoran_fh), GFP_KERNEL);
 if (!fh) {
   dprintk(1,
         KERN_ERR "%s: zoran_open(): allocation of zoran_fh failed\n",
         ZR_DEVNAME(zr));
   return -ENOMEM;
 }
- memset(fh, 0, sizeof(struct zoran_fh));
A SmPL idea: Raise the level of abstraction to semantic patches.

To:

```c
expression x, E1, E2;

- x = kmalloc(E1, E2);
+ x = kzalloc(E1, E2);
  ...
- memset(x, 0, E1);
```

- SmPL = Semantic Patch Language
- Coccinelle applies SmPL semantic patches across a code base.
Over 5500 commits.

44% of the 88 kernel developers who have at least one commit that touches 100 files also have at least one commit that uses Coccinelle.

59 semantic patches in the Linux kernel, usable via make coccicheck.
How did we get here?
Design dimensions

- Expressivity
- Performance
- Correctness guarantees
- Dissemination

Did we make the right decisions?
Original hypothesis: Linux kernel developers will find it easy and convenient to describe needed code changes in terms of fragments of removed and added code.

```c
expression x,E1,E2;
- x = kmalloc(E1,E2);
+ x = kzalloc(E1,E2);
...  
- memset(x, 0, E1);
```
Confrontation with the real world:

- Many language evolutions: C features, metavariable types, etc.
- Position variables.
  - Record and match position of a token.
- Scripting language rules.
  - Original goal: bug finding, eg buffer overflows.
  - Used in practice for error reporting, counting, etc.
expression object;
position p
@@
(

drm_connector_reference@p(object)
|
drm_connector_unreference@p(object)
)

@script:python@

object << r.object;
p << r.p;
@@

msg="WARNING: use get/put helpers to reference and dereference %s" % (object)
coccilib.report.print_report(p[0], msg)
Status: Use of new features

- 3325 commits contain semantic patches.
- 18% use position variables.
- 5% use scripts.
- 43% of the semantic patches using position variables or scripts are from outside the Coccinelle team.
- All 59 semantic patches in the Linux kernel use both.
Coccinelle design: performance

Goal: Be usable on a typical developer laptop.

Target code base: 5MLOC in Feb 2007, 16.5MLOC in Jan 2018.

Original design choices:

- Intraprocedural, one file at a time.
- Process only .c files, by default.
- Include only local or same-named headers, by default.
- No macro expansion, instead use heuristics to parse macro uses.
- Provide best-effort type inference, but no other program analysis.
Performance evolutions

Confrontation with the real world:

- 1, 5, or 15 MLOC is a lot of code.
- Parsing is slow, because of backtracking heuristics.
Confrontation with the real world:

- 1, 5, or 15 MLOC is a lot of code.
- Parsing is slow, because of backtracking heuristics.

Evolutions:

- Indexing, via glimpse, id-utils.
- Parallelism, via parmap.
Based on the 59 semantic patches in the Linux kernel.
Coccinelle design: correctness guarantees

Ensure that outermost terms are replaced by like outermost terms

@@
expression x,E1,E2,E3;
@@
- x = kmalloc(E1,E2);
+ x = kzalloc(E1,E2);
  ...
- memset(x, 0, E1);

No other correctness guarantees:

- Bug fixes and evolutions may not be semantics preserving.
- Improves expressiveness and performance.
- Rely on developer’s knowledge of the code base and ease of creating and refining semantic patches.
Correctness guarantee evolutions

Confrontation with the real world:

Mostly, developer control over readable rules is good enough.
Show by example:

- **June 1, 2007**: Fix parse errors in kernel code.
- **July 6, 2007**: Irq function evolution
  - Updates in 5 files, in net, atm, and usb
- **July 19, 2007**: kmalloc + memset $\rightarrow$ kzalloc
  - Updates to 166 calls in 146 files.
  - A kernel developer responded “Cool!”.
  - Violated patch-review policy of Linux.
- **July 2008**: Use by a non-Coccinelle developer.
- **October 2008**: Open-source release.
Coccinelle design: dissemination strategy

Show by example:

- **June 1, 2007**: Fix parse errors in kernel code.
- **July 6, 2007**: Irc function evolution
  - Updates in 5 files, in net, atm, and usb
- **July 19, 2007**: kmalloc + memset → kzalloc
  - Updates to 166 calls in 146 files.
  - A kernel developer responded “Cool!”.
  - Violated patch-review policy of Linux.
- **July 2008**: Use by a non-Coccinelle developer.
- **October 2008**: Open-source release.

```c
@ rule1 @
typedef irqreturn_t;
@@
static irqreturn_t
fn(int irq, void *dev_id)
{
    ... 
}
@@
identifier rule1.fn;
expression E1, E2, E3;
@@
fn(E1, E2 - ,E3)
```
Dissemination strategy evolutions

Confrontation with the real world:

- Showing by example generated initial interest.
- Organized four workshops: industry participants.
- Presentations at developer conferences: FOSDEM, Linux Plumbers, etc.
- LWN articles by kernel developers.
Impact: Maintainer use

![Graph showing the number of commits for cleanups and bug fixes from 2008 to 2017.](image-url)
Impact: Maintainer use examples

TTY. Remove an unused function argument.

- 11 affected files.

DRM. Eliminate a redundant field in a data structure.

- 54 affected files.

Interrupts. Prepare to remove the irq argument from interrupt handlers, and then remove that argument.

- 188 affected files.
Impact: Some recent commits made using Coccinelle

- Wolfram Sang: tree-wide: simplify getting .drvdta: LKML, April 19, 2018
- Kees Cook: treewide: init_timer() -> setup_timer(): b9eaf1872222
- Deepa Dinamani: vfs: change inode times to use struct timespec64: 95582b008388
59 semantic patches in the Linux kernel with a dedicated make target.
Coccinelle community

25 contributors

- Most from the Coccinelle team, due to use of OCaml and PL concepts.
- Active mailing list (cocci@systeme.lip6.fr).

Availability

- Packaged for many Linux distros.

Use outside Linux

- RIOT, systemd, qemu, zephyr (in progress), etc.
How do you know if the code you have changed has been checked by the compiler?
How do you know if the code you have changed has been checked by the compiler?

- JMake uses heuristics to choose a suitable configuration and mutations to check that changed lines are compiled.
How do you know if the code you have changed has been checked by the compiler?

- JMake uses heuristics to choose a suitable configuration and mutations to check that changed lines are compiled.
- Published in DSN 2017
How do you know if the code you have changed has been checked by the compiler?

• JMake uses heuristics to choose a suitable configuration and mutations to check that changed lines are compiled.

• Published in DSN 2017

• http://jmake-release.gforge.inria.fr
Julia (2011?): Here’s a patch to fix a missing kfree.
Julia (2011?): Here’s a patch to fix a missing kfree.

Helpful kernel maintainer:
That looks OK, but the code should really be using devm_kzalloc
Julia (2011?): Here’s a patch to fix a missing kfree.

Helpful kernel maintainer:
That looks OK, but the code should really be using devm_kzalloc

Julia: What’s that???
Julia (2011?): Here’s a patch to fix a missing kfree.

Helpful kernel maintainer:
That looks OK, but the code should really be using devm_kzalloc

Julia: What’s that???

```
git grep devm_kzalloc???

git log -S devm_kzalloc???
```
**Prequel**: patch queries for searching commit histories.

Query:

@@
@@
```c
- kzalloc
+ devm_kzalloc
(...)
```

Returns the most pertinent commits at the top of the result list.
Prequel for driver backporting:

- Compile driver with the target version.
- Use Gcc-reduce to analyze the error messages and construct patch queries.
- Use Prequel to collect examples of the needed changes.
- Scan through the high-ranked results and figure out how to change the code.
Prequel for driver backporting:

- Compile driver with the target version.
- Use Gcc-reduce to analyze the error messages and construct patch queries.
- Use Prequel to collect examples of the needed changes.
- Scan through the high-ranked results and figure out how to change the code.

Published at USENIX ATC 2017. Tested on 33 drivers with 75% success.

Conclusion

• Initial design decisions mostly remain valid, with some extensions.
  – Take the expertise of the target users into account.
  – Avoid creeping featurism: Do one thing and do it well.

• Tool should be easy to access and install, and easy to use and robust.

• Over 5500 commits in the Linux kernel based on Coccinelle.

• JMake and Prequel inspired by experience with Coccinelle.
Conclusion

- Initial design decisions mostly remain valid, with some extensions.
  - Take the expertise of the target users into account.
  - Avoid creeping featurism: Do one thing and do it well.
- Tool should be easy to access and install, and easy to use and robust.
- Over 5500 commits in the Linux kernel based on Coccinelle.
- JMake and Prequel inspired by experience with Coccinelle.
- Probably, everyone in this room uses some Coccinelle modified code!
Conclusion

- Initial design decisions mostly remain valid, with some extensions.
  - Take the expertise of the target users into account.
  - Avoid creeping featurism: Do one thing and do it well.
- Tool should be easy to access and install, and easy to use and robust.
- Over 5500 commits in the Linux kernel based on Coccinelle.
- JMake and Prequel inspired by experience with Coccinelle.
- Probably, everyone in this room uses some Coccinelle modified code!


Thanks to ANR, Inria, OSADL, Coccinelle users, and many others!