Automating Source Code Evolutions using Coccinelle

Julia Lawall (Inria/LIP6)

Joint work with
Gilles Muller, René Rydhof Hansen,
Nicolas Palix

September 24, 2013

Legacy software: Changing priorities, changing requirements

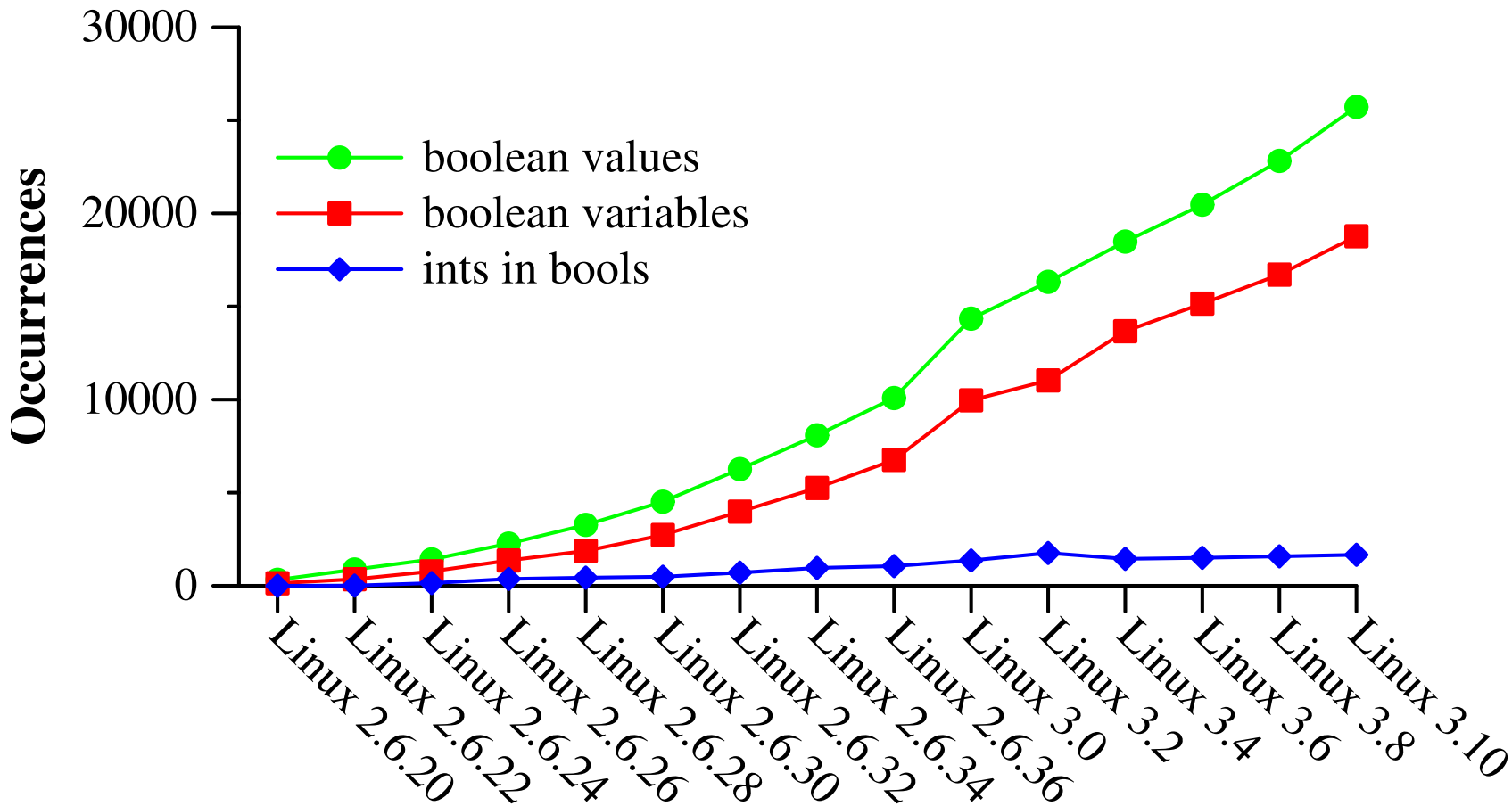
Linux kernel examples:

- Booleans:
  - Use `0` / `1`?
  - Use `true` / `false`?
- Managed memory:
  - `kmalloc` requires `kfree`, `request_irq` requires `free_irq`
  - Devm interface implicitly cleans up allocated resources.

Issues:

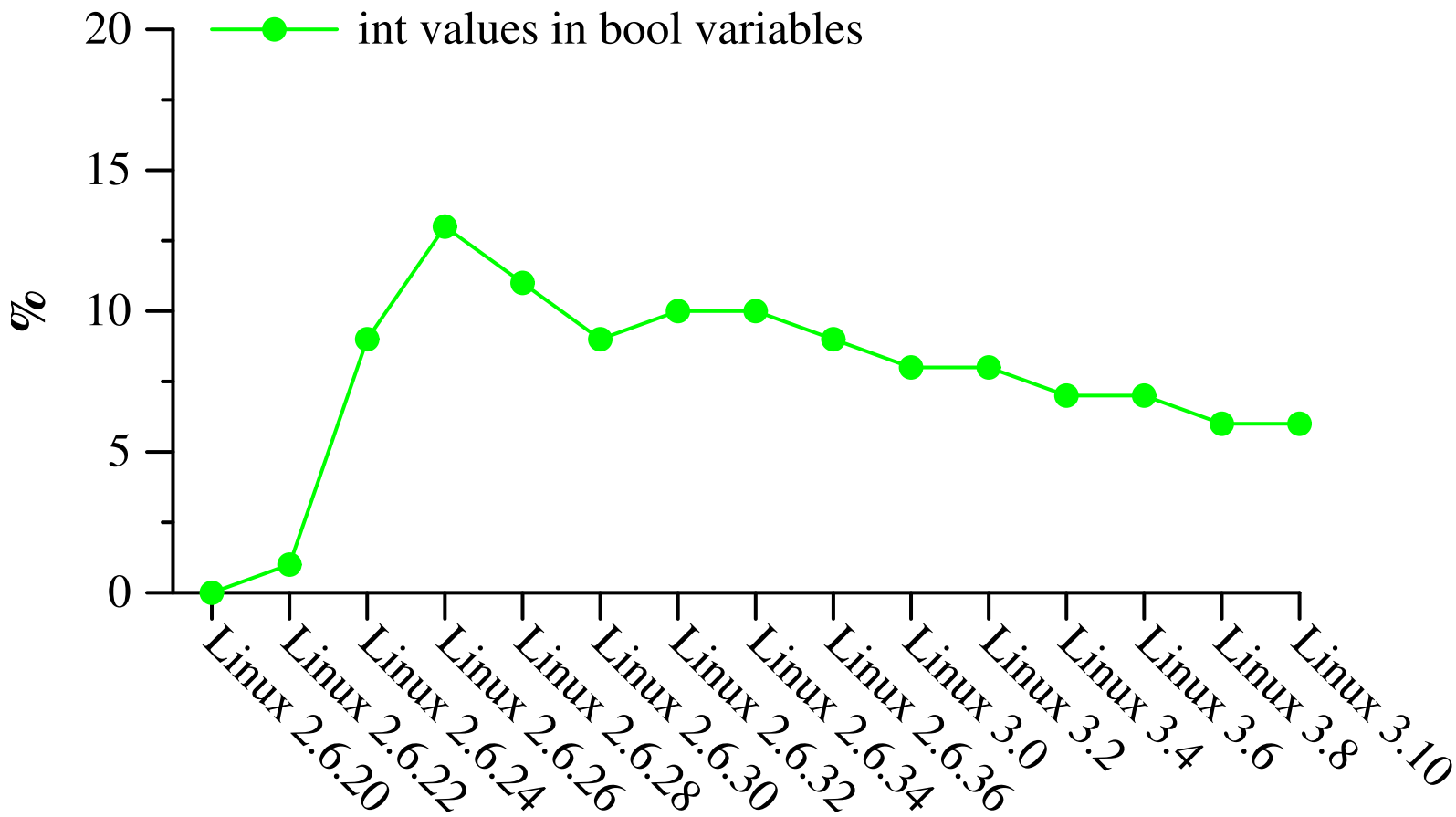
- These changes require pervasive, scattered modifications.
- Nothing forces the changes to be made.
- Developers don't pick up on new coding strategies.

The use of booleans over time



(Roughly every 6 months, February 2007 - June 2013).

The use of booleans over time - rate of bad code



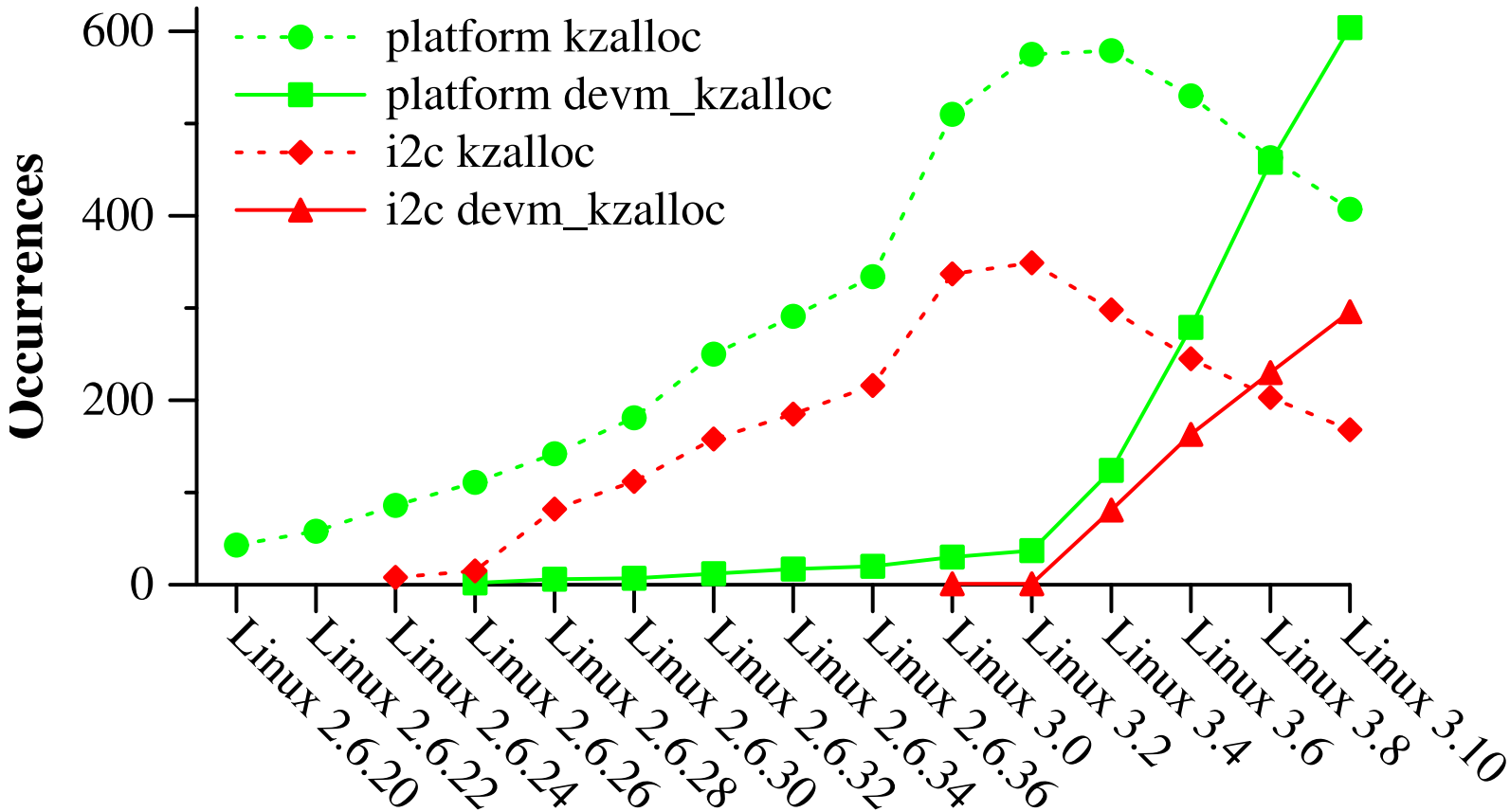
Booleans: A concrete example

Desired property: `bool` variables should be `true` or `false`.

Code fragment:

```
static bool overlapping_resync_write(struct drbd_conf *mdev, ...) {
    struct drbd_peer_request *rs_req;
    bool rv = 0;
    spin_lock_irq(&mdev->tconn->req_lock);
    list_for_each_entry(rs_req, &mdev->sync_ee, w.list) {
        if (overlaps(peer_req->i.sector, peer_req->i.size,
                     rs_req->i.sector, rs_req->i.size)) {
            rv = 1;
            break;
        }
    }
    spin_unlock_irq(&mdev->tconn->req_lock);
    return rv;
}
```

The use of devm functions over time



Devm functions: A concrete example

Desired property (simpler than introducing devm functions):

- A common pattern is `platform_get_resource` followed by `devm_ioremap_resource`.
- `devm_ioremap_resource` does error handling for the `platform_get_resource` value.
- Separate error handling is not needed.

Code fragment:

```
res = platform_get_resource(pdev, IORESOURCE_MEM, 0);
if (!res)
    return -ENODEV;
ahb->regs = devm_ioremap_resource(&pdev->dev, res);
```

How to make these changes automatically and reliably?

Requirements:

- Find relevant code.
  - grep does this...
- Make changes.
  - sed does this...

Problem: Grep and sed don't know about code structure or semantics.

Our approach: Coccinelle

- Static analysis to find patterns in C code.
- Automatic transformation.
- User scriptable, based on patch notation (semantic patches).

http://coccinelle.lip6.fr/

http://coccinellery.org/

Boolean example, part 1

```
@@ -1932,14 +1932,14 @@
static bool overlapping_resync_write(struct drbd_conf *mdev, ...) {
    struct drbd_peer_request *rs_req;
-   bool rv = 0;
+   bool rv = false;
    spin_lock_irq(&mdev->tconn->req_lock);
    list_for_each_entry(rs_req, &mdev->sync_ee, w.list) {
        if (overlaps(peer_req->i.sector, peer_req->i.size,
                     rs_req->i.sector, rs_req->i.size)) {
-           rv = 1;
+           rv = true;
            break;
        }
    }
    spin_unlock_irq(&mdev->tconn->req_lock);
    return rv;
}
```

First step:

- Replace `bool b = 0;` by `bool b = false;`
- Replace `bool b = 1;` by `bool b = true;`

Semantic patch: booleans, part 1

```
@@
identifier b;
@@

bool b =
-       0
+       false
;

@@
identifier b;
@@

bool b =
-       1
+       true
;
```

Semantic patch application: booleans, part 1

Result:

```
@@ -1932,14 +1932,14 @@
static bool overlapping_resync_write(struct drbd_conf *mdev, ...) {
    struct drbd_peer_request *rs_req;
-   bool rv = 0;
+   bool rv = false;
    spin_lock_irq(&mdev->tconn->req_lock);
    list_for_each_entry(rs_req, &mdev->sync_ee, w.list) {
        if (overlaps(peer_req->i.sector, peer_req->i.size,
                     rs_req->i.sector, rs_req->i.size)) {
            rv = 1;
            break;
        }
    }
    spin_unlock_irq(&mdev->tconn->req_lock);
    return rv;
}
```

Affects 187 lines, 124 files in Linux 3.10.

- Fixes the declaration, but not the subsequent assignment...

Boolean example: part 2

Problem:

- We cannot replace every `0` by `false`, and `1` by `true`.
- The assignment must involve a boolean variable.

Solution: Type constraints on metavariables.

```
@@ bool b; @@
b =
-    0
+    false

@@ bool b; @@
b =
-    1
+    true
```

Semantic patch application: booleans, part 2

Result:

```
@@ -1932,14 +1932,14 @@
static bool overlapping_resync_write(struct drbd_conf *mdev, ...) {
    struct drbd_peer_request *rs_req;
-   bool rv = 0;
+   bool rv = false;
    spin_lock_irq(&mdev->tconn->req_lock);
    list_for_each_entry(rs_req, &mdev->sync_ee, w.list) {
        if (overlaps(peer_req->i.sector, peer_req->i.size,
                     rs_req->i.sector, rs_req->i.size)) {
-           rv = 1;
+           rv = true;
            break;
        }
    }
    spin_unlock_irq(&mdev->tconn->req_lock);
    return rv;
}
```

Affects 657 lines, 236 files in Linux 3.10.

Remaining issues

- Function arguments.
- Function return values.
- Booleans used with non-`bool` variables.

Beyond the scope of this talk...

platform_get_resource example, part 1

```
mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
if (!mem) {
    dev_err(&pdev->dev, "no mem resource?");
    return -ENODEV;
}
irq = platform_get_resource(pdev, IORESOURCE_IRQ, 0);
if (!irq) {
    dev_err(&pdev->dev, "no irq resource?");
    return -ENODEV;
}
... // 35 lines of code
dev->base = devm_ioremap_resource(&pdev->dev, mem);
if (IS_ERR(dev->base)) {
    r = PTR_ERR(dev->base);
    goto err_unuse_clocks;
}
```

`platform_get_resource` example, part 1

```
- mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
- if (!mem) {
-   dev_err(&pdev->dev, "no mem resource?");
-   return -ENODEV;
- }
  ...
+ mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
  dev->base = devm_ioremap_resource(&pdev->dev, mem);
  if (IS_ERR(dev->base)) {
      r = PTR_ERR(dev->base);
      goto err_unuse_clocks;
  }
```

Problem: Need both `platform_get_resource` and `devm_ioremap_resource`.

- Separated by arbitrary code fragments.

Semantic patch: `platform_get_resource`, part 1

Copy the code fragment, from the first line to the last line

```
@@


@@

  mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
  if (!mem) {
      dev_err(&pdev->dev, "no mem resource?");
      return -ENODEV;
  }
  irq = platform_get_resource(pdev, IORESOURCE_IRQ, 0);
  if (!irq) {
      dev_err(&pdev->dev, "no irq resource?");
      return -ENODEV;
  }
  ...
  dev->base = devm_ioremap_resource(&pdev->dev, mem);
```

Semantic patch: `platform_get_resource`, part 1

Drop the irrelevant parts

```
@@


@@

  mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
  if (!mem) {
      dev_err(&pdev->dev, "no mem resource?");
      return -ENODEV;
  }




  ...
  dev->base = devm_ioremap_resource(&pdev->dev, mem);
```

Semantic patch: `platform_get_resource`, part 1

Abstract over irrelevant details

```
@@
expression mem, pdev, n, e;
statement S;
@@

  mem = platform_get_resource(pdev, IORESOURCE_MEM, n);
  if (!mem) S

  ...
  e          = devm_ioremap_resource(&pdev->dev, mem);
```

Semantic patch: `platform_get_resource`, part 1

Introduce transformations

```
@@
expression mem, pdev, n, e;
statement S;
@@

- mem = platform_get_resource(pdev, IORESOURCE_MEM, n);
- if (!mem) S
```

```
  ...
+ mem = platform_get_resource(pdev, IORESOURCE_MEM, n);
  e          = devm_ioremap_resource(&pdev->dev, mem);
```

Semantic patch: `platform_get_resource`, part 1

Add sanity checks

```
@@
expression mem, pdev, n, e;
statement S;
@@

- mem = platform_get_resource(pdev, IORESOURCE_MEM, n);
- if (!mem) S
  ... when != mem
+ mem = platform_get_resource(pdev, IORESOURCE_MEM, n);
  e          = devm_ioremap_resource(&pdev->dev, mem);
```

Semantic patch application: `platform_get_resource`, part 1

Result:

```
- mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
- if (!mem) {
-     dev_err(&pdev->dev, "no mem resource?");
-     return -ENODEV;
- }
  irq = platform_get_resource(pdev, IORESOURCE_IRQ, 0);
  if (!irq) {
      dev_err(&pdev->dev, "no irq resource?");
      return -ENODEV;
  }
  ...
+ mem = platform_get_resource(pdev, IORESOURCE_MEM, 0);
  dev->base = devm_ioremap_resource(&pdev->dev, mem);
  if (IS_ERR(dev->base)) {
      r = PTR_ERR(dev->base);
      goto err_unuse_clocks;
  }
```

Affects 42 files in Linux 3.10.

`platform_get_resource` example, part 2

Problem: Some calls are missed because `&pdev->dev` is renamed.

Code fragment:

```
struct device *dev = &pdev->dev;
void __iomem *addr = NULL;
struct stmmac_priv *priv = NULL;
struct plat_stmmacenet_data *plat_dat = NULL;
const char *mac = NULL;

res = platform_get_resource(pdev, IORESOURCE_MEM, 0);
if (!res)
    return -ENODEV;

addr = devm_ioremap_resource(dev, res);
if (IS_ERR(addr))
    return PTR_ERR(addr);
```

Semantic patch: `platform_get_resource`, part 2

```
@@
expression res, pdev, n, e, x, e1, e2;
statement S;
@@

  x = &pdev->dev
  ... when != x = e1
- res = platform_get_resource(pdev, IORESOURCE_MEM, n);
- if (!res) S
  ... when != mem
      when != x = e2
+ res = platform_get_resource(pdev, IORESOURCE_MEM, n);
  e = devm_ioremap_resource(x,res);
```

Result: Updates 4 more files in Linux-3.10.

Summary

We have seen the need for:

- Search and replace for atomic code fragments.
  - Declaration and initialization of boolean variables.
- Search and replace using type information.
  - Assignments of boolean variables.
- Search and replace for scattered code fragments
  - `platform_get_resource` somewhere followed by `devm_iomap_resource`.
- Taking renamings into account.
  - `&pdev->dev` vs `dev`.

Conclusion

Coccinelle

- Define and perform arbitrary transformations across a code base.

Reasonable performance in most cases.

- Boolean example takes 7 minutes on a 4 core laptop.

Impact:

- Over 1000 patches in Linux based on Coccinelle.
- Over 40 semantic patches in linux/scripts/coccinelle.

`http://coccinelle.lip6.fr/`
`http://coccinellery.org/`