SmPL: A Domain-Specific Language for Specifying Collateral Evolutions in Linux Device Drivers

Yoann Padioleau
Ecole des Mines de Nantes

Julia Lawall, DIKU, University of Copenhagen
Gilles Muller, Ecole des Mines de Nantes

ERCIM workshop on software evolution, Lille, April 2006

The collateral evolution problem

- Libraries change.
- These changes may affect the library interfaces.
- Dependent code must be updated accordingly.

→ Evolutions in generic library entails modifications,
i.e. collateral evolutions in library clients.

Our target: Linux device drivers.

An example

Interface change: Drivers "proc_info" functions shouldn't use the functions `scsi_host_hn_get` and `scsi_host_put`.

```
static int usb_storage_proc_info (char *buffer, char **start, off_t offset,
          int length, int hostno, int inout) {
  struct us_data *us;
  char *pos = buffer;
  struct Scsi_Host *hostptr;
  unsigned long f;

  if (inout) return length;
  hostptr = scsi_host_hn_get(hostno);
  if (!hostptr) { return -ESRCH; }
  us = (struct us_data*)hostptr->hostdata[0];
  if (!us) {
    scsi_host_put(hostptr);
    return -ESRCH;
  }
  <SOME CODE OMMITED>
  scsi_host_put(hostptr);
  <SOME CODE OMMITED>
 }
```

An example

Interface change: Drivers "proc_info" functions shouldn't use the functions `scsi_host_hn_get` and `scsi_host_put`.

```
static int usb_storage_proc_info (char *buffer, char **start, off_t offset,
          int length, int hostno, int inout) {
  struct us_data *us;
  char *pos = buffer;
  struct Scsi_Host *hostptr;
  unsigned long f;

  if (inout) return length;
  hostptr = scsi_host_hn_get(hostno);
  if (!hostptr) { return -ESRCH; }
  us = (struct us_data*)hostptr->hostdata[0];
  if (!us) {
    scsi_host_put(hostptr);
    return -ESRCH;
  }
  <SOME CODE OMMITED>
  scsi_host_put(hostptr);
  <SOME CODE OMMITED>
 }
```

A corresponding patch file

```
--- a/drivers/usb/storage/scsiglue.c Sat Jun 14 12:18:55 2003
+++ b/drivers/usb/storage/scsiglue.c Sat Jun 14 12:18:55 2003
@@ -264,33 +300,21 @@
-static int usb_storage_proc_info (
+static int usb_storage_proc_info (struct Scsi_Host *hostptr,
                char *buffer, char **start, off_t offset,
-               int length, int hostno, int inout) {
+               int length, int inout) {
    struct us_data *us;
    char *pos = buffer;
-   struct Scsi_Host *hostptr;
    unsigned long f;
    if (inout) return length;

-   hostptr = scsi_host_hn_get(hostno);
-   if (!hostptr) {
-           return -ESRCH;
-   }
    us = (struct us_data*)hostptr->hostdata[0];
    if (!us) {
-           scsi_host_put(hostptr);
            return -ESRCH;
    }
@@ -318,9 +342,6 @@
-   scsi_host_put(hostptr);
```

A corresponding patch file

```
--- a/drivers/usb/storage/scsiglue.c Sat Jun 14 12:18:55 2003
+++ b/drivers/usb/storage/scsiglue.c Sat Jun 14 12:18:55 2003
@@ -264,33 +300,21 @@
-static int usb_storage_proc_info (
+static int usb_storage_proc_info (struct Scsi_Host *hostptr,
                char *buffer, char **start, off_t offset,
-               int length, int hostno, int inout) {
+               int length, int inout) {
    struct us_data *us;
    char *pos = buffer;
-   struct Scsi_Host *hostptr;
    unsigned long f;
    if (inout) return length;

-   hostptr = scsi_host_hn_get(hostno);
-   if (!hostptr) {
-           return -ESRCH;
-   }
    us = (struct us_data*)hostptr->hostdata[0];
    if (!us) {
-           scsi_host_put(hostptr);
            return -ESRCH;
    }
@@ -318,9 +342,6 @@
-   scsi_host_put(hostptr);
```

A corresponding patch file

```
--- a/drivers/usb/storage/scsiglue.c Sat Jun 14 12:18:55 2003
+++ b/drivers/usb/storage/scsiglue.c Sat Jun 14 12:18:55 2003
@@ -264,33 +300,21 @@
-static int usb_storage_proc_info (
+static int usb_storage_proc_info (struct Scsi_Host *hostptr,
                char *buffer, char **start, off_t offset,
-               int length, int hostno, int inout) {
+               int length, int inout) {
    struct us_data *us;
    char *pos = buffer;
-   struct Scsi_Host *hostptr;
    unsigned long f;
    if (inout) return length;

-   hostptr = scsi_host_hn_get(hostno);
-   if (!hostptr) {
-           return -ESRCH;
-   }
    us = (struct us_data*)hostptr->hostdata[0];
    if (!us) {
-           scsi_host_put(hostptr);
            return -ESRCH;
    }
@@ -318,9 +342,6 @@
-   scsi_host_put(hostptr);
```

Motivations

Collateral evolutions in Linux are mostly done manually (with a basic editor and the help of `grep`).

Some collateral evolutions can involve:

- hundreds of files
- thousands of code sites

Some collateral evolutions are quite complex.

$\rightarrow$ Collateral evolutions in Linux is thus time consuming and error prone.

$\rightarrow$ We propose a language to specify easily and automate such program transformations.

Proposal for a semantic patch language (SmPL)

```
   proc_info_func (
+       struct Scsi_Host *hostptr,
        char *buffer, char **start, off_t offset, int length,
-       int hostno,
        int inout) {
```

Proposal for a semantic patch language (SmPL)

```
@@
identifier buffer, start, offset, length, inout, hostptr, hostno;
@@
   proc_info_func (
+       struct Scsi_Host *hostptr,
        char *buffer, char **start, off_t offset, int length,
-       int hostno,
        int inout) {
```

Proposal for a semantic patch language (SmPL)

```
@@
struct SHT sht;
local function proc_info_func;
@@
   sht.proc_info = &proc_info_func;

@@
identifier buffer, start, offset, length, inout, hostptr, hostno;
@@
   proc_info_func (
+       struct Scsi_Host *hostptr,
        char *buffer, char **start, off_t offset, int length,
-       int hostno,
        int inout) {
```

Proposal for a semantic patch language (SmPL)

```
@@
struct SHT sht;
local function proc_info_func;
@@
   sht.proc_info = &proc_info_func;

@@
identifier buffer, start, offset, length, inout, hostptr, hostno;
@@
   proc_info_func (
+       struct Scsi_Host *hostptr,
        char *buffer, char **start, off_t offset, int length,
-       int hostno,
        int inout) {
    ...
-   struct Scsi_Host *hostptr;
    ...
-   hostptr = scsi_host_hn_get(hostno);
    ...
-   if (!hostptr) { ... }
    ...
-   scsi_host_put(hostptr);
    ...
  }
```

Summary

A single small semantic patch can modify hundreds of files, at thousands of code sites.

This is because the features of SmPL make a semantic patch generic by abstracting away the specific details at each code site:

- differences in spacing, indentation, and comments.
- choice of the names given to variables (use of metavariables).
- different way to sequence instructions in C (control-flow oriented rather than AST oriented).
- other variations in coding style (use of isomorphisms).