The Mythical Matched Modules
Or: overcoming the tyranny of inflexible software construction.

Stephen Kell
Stephen.Kell@cl.cam.ac.uk

Computer Laboratory
UNIVERSITY OF CAMBRIDGE
There’s something about software... Software is expensive and *inflexible*.

Tools assume:
- ground-up
- perfect fit
- don’t change
- never replaced

Reality: none of the above!
What it means in practice

Tasks dealing with change are notoriously hard work.

- porting
- wrapping

Language innovations are notoriously slow to be adopted

- each language is its own silo

Usual solution is “don’t do that!”

- rewrite from scratch
- single-language programming
The idea in one slide

Parnas pioneered *information hiding*.

- interface changes are painful, so . . .
- keep interfaces *minimal*, to avoid change

But interfaces *do* change. What can be done?

- radically separate integration: *interface hiding*
  - Error: “import” statement deprecated
- specialized tool support for integration
  - complementary languages (Cake is one example)
decoder = mpeg2_init();

info = mpeg2_info(decoder);

do {
    state = mpeg2_parse(decoder);
    switch (state) {
        case STATE_BUFFER:
            size = fread(buffer, /* ... */);
            mpeg2_buffer(decoder, buffer, /* ... */);
            break;

        case STATE_SLICE: case /* ... */:
            for (i = 0; i < seq->luma_height; i++)
                fwrite(info->display_fbuf->/* ... */);
            break;
    }
} while (size);

mpeg2_close(decoder);
avcodec_init(); av_register_all();

codec = avcodec_find_decoder(c->codec_id);
ap->time_base = (AVRational){1, 25};
ap->pix_fmt = PIX_FMT_NONE;
err = av_find_stream_info(ic);
for (i = 0; i < ic->nb_streams; i++) {
    AVCCodecContext *enc = ic->streams[i]->codec;
    if (enc->codec_type == CODEC_TYPE_VIDEO)
        video_index = i;
}
c = ic->streams[video_index]->codec;
for (; ;) { while (pkt->size > 0) {
    picture = avcodec_alloc_frame();
    len = avcodec_decode_video2(c, picture, &got_picture, pkt);
    if (len >= 0 && got_picture) {
        for (i=0; i < c->height; i++) /* for each row */
Separate integration

Hardware (and other domains)
- keep components simple
- ... and composable
- ... and cheap

By separating integration...
- *physically* (modules)
- *notationally* (languages)
Cake: an integration domain

Just one example of an integration domain: Cake.

- declarative language of *interface correspondences*
- complements general-purpose programming languages

```c
derive /* ... */ WholeProgram = link[someClient.o, my_library.so ]
{
    // ...
    someClient.o ↔ my_library.so {
        manipulate_foo(i, d, c_h, data)
        → make_foobaz(i, d, c_h, data, null , {});
        // more correspondences ...
    }
    // more pairwise blocks...
};
```
Conclusions

Interface hiding is a radical “next step” in modularity.

- reduce component complexity
- minimises coupling
- retain composability with many libraries

Integration domains abstract integration

- better notational abstractions
- absorb change; absorb language mismatch; . . .
- incremental adoption is feasible

Thanks for your attention. Any questions?
Revision 546826

Author: thiago
Date: Wed May 31 07:20:26 2006 UTC (3 years, 4 months ago)

Log Message:
Since no objections were raised in kde-core-devel, I am merging the kdelibs4-dbus branch back into trunk. KDELibs compiles, links and installs with this, but obviously all other modules will fail to build. Let the porting commence.

CCMAIL: kde-core-devel@kde.org, kde-buildsystem@kde.org
<table>
<thead>
<tr>
<th></th>
<th>mpeg2</th>
<th>ffmpeg</th>
<th>db</th>
<th>sqlite</th>
</tr>
</thead>
<tbody>
<tr>
<td>API function count</td>
<td>24</td>
<td>151</td>
<td>208</td>
<td>214</td>
</tr>
<tr>
<td>mean signature size</td>
<td>3.3</td>
<td>4.3</td>
<td>2.8</td>
<td>4.3</td>
</tr>
<tr>
<td>API structures count</td>
<td>10</td>
<td>26</td>
<td>25</td>
<td>12</td>
</tr>
<tr>
<td>mean structure size</td>
<td>5.6</td>
<td>19</td>
<td>17</td>
<td>8.0</td>
</tr>
<tr>
<td>slice function count</td>
<td>5</td>
<td>14</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>mean signature size</td>
<td>2.8</td>
<td>3.1</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>slice structures count</td>
<td>3</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>client size (LoC)</td>
<td>71</td>
<td>93</td>
<td>51</td>
<td>75</td>
</tr>
<tr>
<td>client API calls</td>
<td>5</td>
<td>14</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>helper calls</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>mean call size</td>
<td>3.4</td>
<td>3.2</td>
<td>4.9</td>
<td>3.9</td>
</tr>
</tbody>
</table>