Executable Texts

SHOT, Washington, DC

stuart mawler
Virginia Tech, Northern Virginia Campus
2007/10/19
Background

- **Usual view**
  - Software running end product
    - e.g., Microsoft Word or PowerPoint

- **Alternative view – Executable Texts**
  - Software source code as document, manuscript, corpus, or text
  - Consumed among communities of programmers
  - Continual historical archive, written and re-written
    - Programmer acts like “Talmudic” scholar

- **This presentation**
  - Investigates two sub-communities
    - Linux Open Source Programmers working on the operating system kernel in C
    - Corporate programmers working on a core business application in COBOL
What I Found...

- “Real constraints”
  - The artifacts are impacted by practical considerations outside of the programming task

- Social roles of executable texts
  - What the common laborer is doing, rather than those in charge
    - The lives and intentions of actual people are encapsulated in the comments
      - Even when created for “capitalist” ends, the personalities of the programmers show through

- “Programming archaeology” / Micro-histories
  - Viewed through the modifications to a program or set of programs
    - Both “explicit” and “implicit”
    - A single program is a form of virtual archaeology, peeling back the layers of modifications by various people for various projects
Linguistic Requirements

- How to define a comment...
  - in COBOL (corporate program)

```plaintext
00892 *----SET ADDRESS OF MESSAGE MAIN HEADER.                          PRGMNBR1
00893 *                                                                          PRGMNBR1
```

- in C++ (online programming guide)

```plaintext
//=================================================================================
//Development By : Jigar Mehta
//Date : [ & now() & ]
//=================================================================================
```

- in C (Linux kernel)

```plaintext
/* Arch-specific enabling code. */
```
Technical Environment

- Different visual constraints

Source: www.dataconnection.com/sna/images/snapix3270.gif
Cultural Constraints and Norms

- Accepted practices
  - Normative purpose for the comment
    - Power structure driving the norms
  - Content
  - Visual Impact

Purpose:
- Identify “who” & “when”

Power:
- Developed by a manager

Content:
- Nothing about the code

Visual:
- Easy to see when scanning

---

/* Arch-specific enabling code. */ (cpu.c)

//Development By : Jigar Mehta
//Date : [ & now() & ]
//============================================================//

#ifndef CONFIG_HOTPLUG_CPU
#endif /*CONFIG_HOTPLUG_CPU*/ (cpu.c)
"Good" Comments

/* Comments have both Form and Function, which exist along a continuum from Normative to Identity-oriented. */

/* Each comment can be plotted against these two axes. */
Good Comments – By Example

Corporate Sample

- Written in COBOL
- Tells what is happening
- What has been changed
- How it was changed
- Does not tell “who” or “when”
- Does say “why”, subtly with “PSR3”
  - Shows much context, if you know the context...

Good, not “historical”:

#include /*CONFIG_HOTPLUG_CPU*/ (cpu.c)
Explicit History

- A snippet of the “change log” at the top of a COBOL program

<table>
<thead>
<tr>
<th></th>
<th>LEVEL</th>
<th>DATE</th>
<th>CHANGED BY</th>
<th>CHANGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0013</td>
<td></td>
<td></td>
<td></td>
<td>Change Log</td>
</tr>
<tr>
<td>0014</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0015</td>
<td>LEVEL</td>
<td>DATE</td>
<td>CHANGED BY</td>
<td>CHANGE</td>
</tr>
<tr>
<td>0016</td>
<td>------</td>
<td>------------</td>
<td>------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>0017</td>
<td></td>
<td>8/10/94</td>
<td>B.K.</td>
<td>Deleted Change Log Entries</td>
</tr>
<tr>
<td>0018</td>
<td>FO</td>
<td>06-21-93</td>
<td>J.C.</td>
<td>Add Security To AB Cancel Reason</td>
</tr>
<tr>
<td>0019</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0020</td>
<td></td>
<td>07-27-93</td>
<td>R.P.</td>
<td>Obtain Language Literals Reason</td>
</tr>
<tr>
<td>0021</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0022</td>
<td></td>
<td></td>
<td></td>
<td>From Decode File Instead Of Hard-Coded Table</td>
</tr>
<tr>
<td>0023</td>
<td></td>
<td>08/02/93</td>
<td>B. H.</td>
<td>Changed Customer First Transaction Code Value</td>
</tr>
<tr>
<td>0024</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0025</td>
<td></td>
<td></td>
<td></td>
<td>From 'PI' to 'RP'</td>
</tr>
<tr>
<td>0026</td>
<td></td>
<td>08/20/93</td>
<td>P. S.</td>
<td>Add Call To Subroutine</td>
</tr>
<tr>
<td>0027</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0028</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0029</td>
<td></td>
<td>10/02/93</td>
<td>R. P.</td>
<td>Added Code To Update BTN</td>
</tr>
<tr>
<td>0030</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0031</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0032</td>
<td></td>
<td>10/05/93</td>
<td>V. S.</td>
<td>Added Logic To Allow Update Of Account</td>
</tr>
<tr>
<td>0033</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Explicit History

- A snippet of the “change log” at the top of a Linux Kernel program: sched.c

```c
/*
 *  kernel/sched.c
 *  Kernel scheduler and related syscalls
 *  Copyright (C) 1991-2002  Linus Torvalds
 *
 *  1996-12-23  Modified by Dave Grothe to fix bugs in semaphores and
 *               make semaphores SMP safe
 *  1998-11-19  Implemented schedule_timeout() and related stuff
 *               by Andrea Arcangeli
 *  2002-01-04  New ultra-scalable O(1) scheduler by Ingo Molnar:
 *               hybrid priority-list and round-robin design with
 *               an array-switch method of distributing timeslices
 *               and per-CPU runqueues. Cleanups and useful suggestions
 *               by Davide Libenzi, preemptible kernel bits by Robert Love.
 *  2003-09-03  Interactivity tuning by Con Kolivas.
 *  2004-04-02  Scheduler domains code by Nick Piggin
 */
```
Implicit History

- This example also tells something about programming style, since “structured” had to be “invented” at some point.
  - After the “invention” of structured programming, all that went before it became “spaghetti” and this name also came to apply to improperly done code.
  - Hence, the new methodology both retroactive and proactively made all other approaches “wrong”.

06073 * THAT CONCLUDES THE STRUCTURED COBOL PORTION OF THIS PGM...
06074 * RETURN TO SPAGHETTI CODE!
06075 GO TO 3159-CONTINUE-SPAGHETTI.
Implicit History

* Where “programming archaeology” comes into play:

```assembly
01188 ** MAC 07/17/95 METEOR 2QTR - BEGIN  FOprogm1
01189 10 COM-FOSLSVC1-FLAG          PIC X(01).  FOprogm1
01190  88 GOTO-FOSLSVC1             VALUE 'Y'.  FOprogm1
01191 ** MAC 07/17/95 METEOR 2QTR - END  FOprogm1
01192 10 COM-HOLD-CANCEL-REASON       PIC X(02).  FOprogm1
01193  88 COM-FRAUD-CANCEL-REASON     VALUES 'BA' 'BB'    FOprogm1
01194                       'BD' 'BE'.   FOprogm1
01195 10 COM-SKIP-FLAG-ST            PIC X.        FOprogm1
01196 10 SKIP-FLAG-SW-OTHER          PIC X.        FOprogm1
01197 ** 03/10/95 MC - BEGIN         FOprogm1
01198 10 COM-PA-TELCO-ID             PIC X(04).  FOprogm1
01199 10 COM-CX33-ANI-READ-KEY.      FOprogm1
01200 15 COM-ANI-CUSTOMER-ID-T        PIC X(08).  FOprogm1
01201 15 COM-ANI-PHONE-NBR-T          PIC X(10).  FOprogm1
01202 15 COM-ANI-STATUS-T            PIC X(01).  FOprogm1
01203 ** 03/10/95 MC - END            FOprogm1
01204 ** MAC 07/17/95 METEOR 2QTR - BEGIN  FOprogm1
01205 10 COM-ACN-PRD-DATE             PIC X(08).  FOprogm1
01206 10 COM-ACN-PRD-IND              PIC S9(09) COMP.  FOprogm1
01207 10 COM-PREV-ACN-PRD-IND         PIC S9(09) COMP.  FOprogm1
01208 ** MAC 07/17/95 METEOR 2QTR - END  FOprogm1
01209 ** JHK 11/10/01 SB LOCAL - BEGIN.  FOprogm1
01210 10 COM-BUS-SEG-IND              PIC X(1).        FOprogm1
01211 ** JHK 11/10/01 SB LOCAL - END.   FOprogm1
```
Implicit History

- More “programming archaeology”
  - Layers of edits

```plaintext
02871 ** MAC 07/17/95 METEOR 2QTR - BEGIN
02872    MOVE A-ACN-PRD-DATE OF RECORD
02873    TO COM-ACN-PRD-DATE.
02874 *PS 07/07/97 COM-ACN-PRD-IND NO LONGER NEEDED, SET TO ZERO AND USE
02875 *AS FLAG TO INDICATE IF 500 OR PAGER ON FB-SERVICE (SEE GET-METEOR
02876    MOVE 0
02877 *    TO COM-ACN-PRD-IND.
02878 *    TO COM-ACN-PRD-IND.
02879 *PS 07/07/97 END
02880    MOVE A-PREV-ACN-PRD-IND OF RECORD
02881    TO COM-PREV-ACN-PRD-IND.
02882 ** MAC 07/17/95 METEOR 2QTR - END
```
Implicit History

- Much less prevalent in Linux Kernel, an atypical example:

```c
/*
 * This needs some heavy checking ...
 * I just haven't the stomach for it. I also don't fully
 * understand sessions/pgrp etc. Let somebody who does explain it.
 *
 * OK, I think I have the protection semantics right.... this is really
 * only important on a multi-user system anyway, to make sure one user
 * can't send a signal to a process owned by another. -TYT, 12/12/91
 *
 * Auch. Had to add the 'did_exec' flag to conform completely to POSIX.
 * LBT 04.03.94
 */ (sys.c)
```

- This example is much more typical of longer comments

```c
/*
 * 'User priority' is the nice value converted to something we
 * can work with better when scaling various scheduler parameters,
 * it's a [ 0 ... 39 ] range.
 */

#define USER_PRIO(p) ((p)-MAX_RT_PRIO)
#define TASK_USER_PRIO(p) USER_PRIO((p)->static_prio)
#define MAX_USER_PRIO (USER_PRIO(MAX_PRIO))
```

2007/10/19 mawler - Executable Texts
Technology History within Comments

- Identity-orientated comments: informative in other ways
  - The Commentator – a satirical faux comment generator
Technology History within Comments

- Identity-oriented comments: informative in other ways
  - The Commentator – a satirical faux comment generator

/* okay, finally we are ready to take the important step of summing the integer elements of a. I've researched the most efficient algorithm and settled on this one, presented by Knuth. I don't quite agree with his reasoning but the algorithm is sound (did I tell you about the cheque I got from Knuth? no? It was back in my uni days when I was writing my thesis (youngest ever accepted into the program) in TeX on the PDP-10. I just couldn't get it to format my differential equations properly, and a quick look under the hood...*/
Conclusion

- Comments are both normative & identity-oriented
  - Structure (form & function)
    - Reflects & shapes programmer community & personal identity
    - Reflects levels of association with the machine
  - Maintains historical continuity
    - Continuous historical commentary
    - Whether normative or identity-oriented
    - For both community norms and knowledge
- Much more prevalent in Corporate sample than in Linux kernel
  - Some normative differences
- Extremely valuable research archive for research purposes
History?

- “Lie in the comments”
  - Remember that comments are not tied to the code
  - They can be wrong