Computer Source Code In Litigation

Andrew Schulman
SoftwareLitigationConsulting.com / DisputeSoft.com

Outline of class

1. What is source code?
2. Why should attorneys care? What can you do with it in litigation?
3. Types of source code, and some important distinctions
4. Timeline of source-code use in litigation
5. Discovery & protective orders (POs)
6. Experts & source-code examination skills & methodology
7. How source code relates to computer forensics, e-discovery, etc.
8. Some cases
9. Trends & take-aways

1. What is source code?

a. Briefly, a human-readable form of computer software/instructions
b. Some examples of what source code looks like
c. Some examples of what source code does NOT look like
d. Definitions of source code: © statute, FDA regulations
e. Case-specific definitions of source code e.g. in protective orders (POs)
f. Imperfect analogies to software: blueprint, recipe, piano roll
g. Why it's called "code": instructions, not Capt. Marvel Decoder Ring
h. Code as a special (operable) kind of text or document
i. "The Crown Jewels" -- embodiment of organization's IP and of its practices/policies

1. What is source code?: Examples

a. SHOW some C code of mine -- mkndx.c to index src lines
b. SHOW Covid-19 contact-tracing app code
c. SHOW ProLaw CLE blurb graphic -- yep, this is source code
d. SHOW some lines of code from large open-source collection
e. SHOW something that is NOT code -- data/output
f. SHOW an example of software but NOT source code (an important distinction) -- dump of binary; note binary contains strings not just 1s & 0s

C source code -- from mkndx.c to index source

```
    // if (! f) fail("can't open file"); // no, keep going
    if (! f) { fprintf(stderr, "\nCan't open <%s>\n\n", fname); return 0; }
    buf = alloc(10240);

    // TODO: need to create temp hashtab, so can do 1 entry per file
    // TODO: maybe change MakeHashTab to take param with tab_size, store

    while (fgets(buf, 10239, f))
    {
        char *s = buf;
        buf[strlen(buf)-1] = '\0'; // remove \n
        while (*s == ' ' || *s == '\t') s++; // remove leading spaces
        if (*s)
        {
            cnt++;
            SetHashTab(strings, s, fnum); // will keep count
        }
    }
    fclose(f);
    free(buf);
    return 1;
}

main(int argc, char *argv[])
{
    unsigned i;
```

```
void insert(NODE **list, char *s, unsigned val)
{
DE *node;
DE *l2 = *list;
ile (l2)

  if (strcmp(l2->s, s) == 0) // could do case-inse
  {
        l2->val = val; // took 1 hr. to find bug
        return; // already in list
  }
  else
```

```
void set(NODE **hashtab, char *s, unsigned val)
{
    insert(&hashtab[HASH(s)], s, val);
}

void SetHashTab(HASHTAB hashtab, char *s, unsigned val)
{
    set((NODE**) hashtab, s, val);
}

int get(NODE **hashtab, char *s)
{
   return find(hashtab[HASH(s)], s);
}
```

Example: Singapore COVID-19 contact tracing app “TraceTogether”, “BlueTrace” written in Java/Kotlin)

```
inner class BleScanCallback : ScanCallback() {

    private val TAG = "BleScanCallback"

    private fun processScanResult(scanResult: ScanResult?) {

        scanResult?.let { result ->
            val device = result.device
            var rssi = result.rssi // get RSSI value

            var txPower: Int? = null

            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                txPower = result.txPower
                if (txPower == 127) {
                    txPower = null
                }
            }

            var manuData: ByteArray =
                scanResult.scanRecord?.getManufacturerSpecificData(1023) ?: "N.A".toByte
            var manuString = String(manuData, Charsets.UTF_8)
```

Source code example (JavaScript; comments)

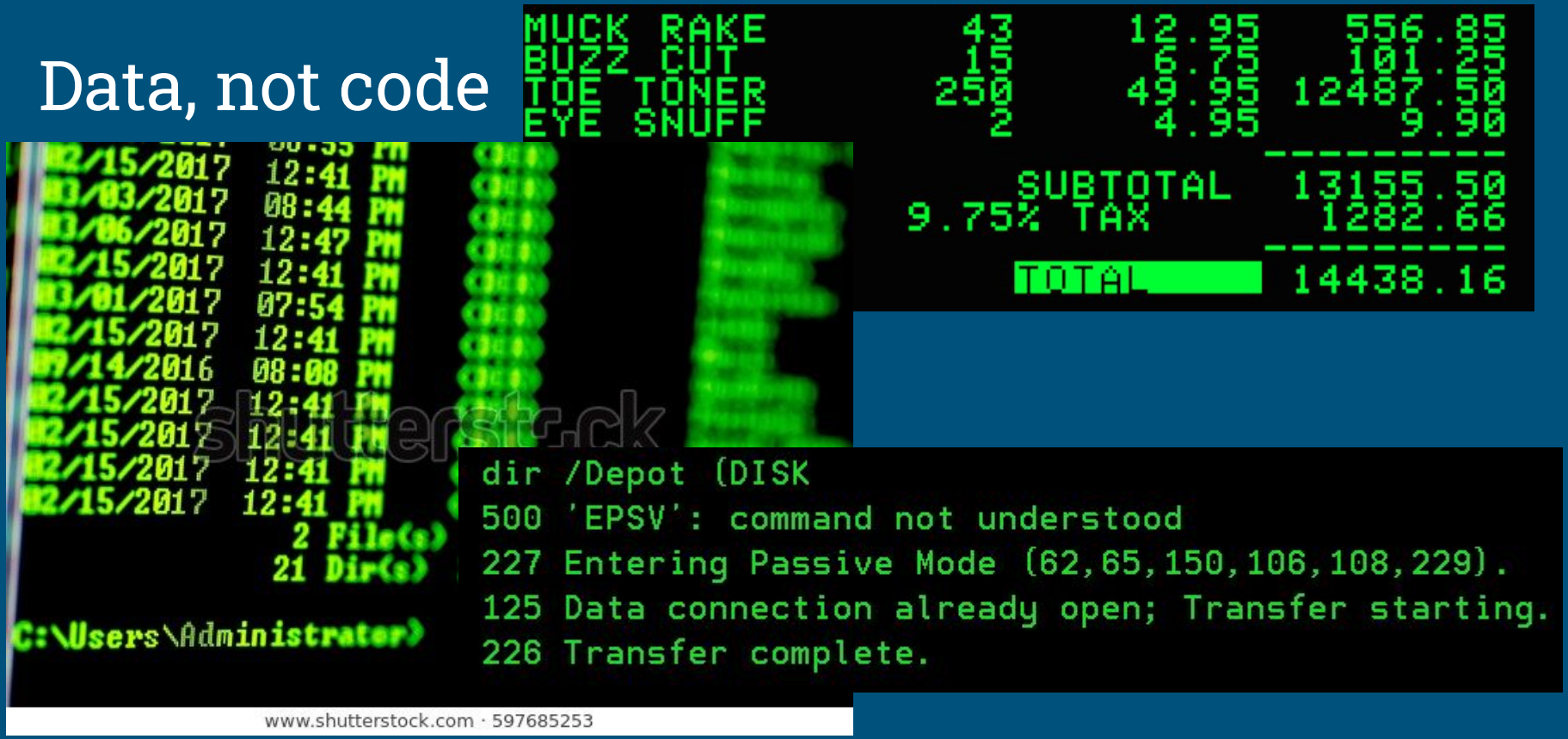
src/contrib/highlight.js

```
@@ -12,6 +12,8 @@ var highlight = function($element, pattern) {

        var highlight = function(node) {
                var skip = 0;
+               // Wrap matching part of text node with highlighting <span>, e.
+               // Soccer  ->  <span class="highlight">Soc</span>cer  for regex
                if (node.nodeType === 3) {
                        var pos = node.data.search(regex);
                        if (pos >= 0 && node.data.length > 0) {
@@ -25,7 +27,10 @@ var highlight = function($element, pattern) {
                                middlebit.parentNode.replaceChild(spannode, mid
                                skip = 1;
                        }
-               } else if (node.nodeType === 1 && node.childNodes && !/(script|
```

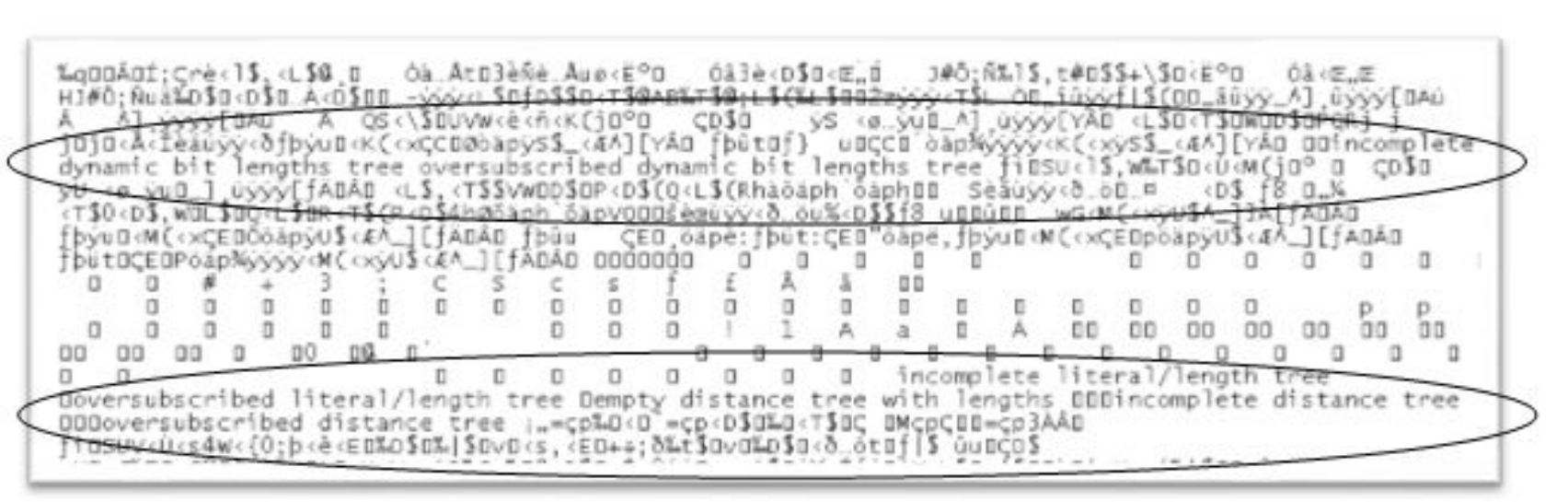
Some lines of code from large open-source set

```
var engineDataTable = "id INTEGER PRIMARY KEY, engineid STRING, name STRING, value STRING";
/* #define RFC2247_ATTR_TYPE 0x09, 0x92, 0x26, 0xf5, 0x98, 0x1e, 0x64, 0x1 this is WRONG! */
virtual void foo (int a, int b, int c, int d) { A::foo (a, b, c, d); D::A::foo (a, b, c, d); }
_Sat unsigned TYPE NAME ## sat_u_sub = (_Sat unsigned TYPE)0.1u ## POSTFIX - 0.2u ## POSTFIX; \
{ "http://a/b/c/d;p?q#f", "get?baseRef=viewcert.jpg", "http://a/b/c/get?baseRef=viewcert.jpg" },
String input = "ecologi-\r\ncal devel-\r\n\r\nop compre-\u0009hensive-hands-on and ecologi-\ncal";
and PDO::ATTR_EMULATE_PREPARES overrules the other, PDO::MYSQL_ATTR_DIRECT_QUERY should be off\n");
run('ant -Dversion=%s -Dspecversion=%s -Dgpg.key=%s prepare-release' % (version, version, gpgKeyID))
static const int kPreExclusion = 0x0040; // IMG, OBJECT, APPLET, BIG, SMALL, SUB, SUP, FONT, BASEFONT
for (x = 0; x < sizeof(zend_signal_globals->pstorage) / sizeof(*zend_signal_globals->pstorage); ++x) {
{ (void *)"EBG Elektronik Sertifika Hizmet Sa\xC4\x9Flay\xc4\xb1\x63\xc4\xbls\xc4\xb1", (PRUint32)48 },
return new AclEnumerator(this, allowedUsersTable, allowedGroupsTable, deniedUsersTable, deniedGroupsTable);
static int ZEND_FASTCALL zend_fetch_var_address_helper_SPEC_CONST_UNUSED(int type, ZEND_OPCODE_HANDLER_ARGS)
static char *nsapi_dlls[] = { "ns-httpd40.dll", "ns-httpd36.dll", "ns-httpd35.dll", "ns-httpd30.dll", NULL };
else if ((i == 0xF9DC) || ((i >= 0xB9AC) && (i <= 0xBBF4)) || ((i >= 0xE0F0) && (i <= 0xE4E5))) return 0xB9AC;
var args = [[5], [5], [5], [5], [5], [5], [5], [5], [5], [5], [5], let (x = []) (x.length = getMaxArgs() + 1, x)]
#define GID_ROTATE_ANGLE_FROM_ARGUMENT(_arg_) ((((double)(_arg_) / 65535.0) * 4.0 * 3.14159265) - 2.0 * 3.14159265)
static const CLSID CLSID_WinParentalControls = {0xE77CC89B,0x7401,0x4C04,{0x8C,0xED,0x14,0x9D,0xB3,0x5A,0xDD,0x04}};
new TestCase( SECTION, "VAR1 = -0; VAR2= -1; VAR1 -= VAR2; VAR1", 1, eval("VAR1 = -0; VAR2 = -1; VAR1 -= VAR2; VAR1") );
static const int nflags_code_kddi[10] = {0x2549, 0x2546, 0x24c0, 0x2545, 0x2548, 0x2547, 0x2750, 0x254a, 0x24c1, 0x27f7};
if (TSendMail(INI_STR("SMTP"), &tsm_err, &tsm_errmsg, hdr, subject, to, message, NULL, NULL, NULL TSRMLS_CC) == FAILURE) {
String result[] = { "我", "购买", "了", "道具", "和", "服装", "我", "购买", "了", "道具", "和", "服装" };
NS_SCRIPTABLE NS_IMETHOD GetPartial(PRBool *aPartial) { return !_to ? NS_ERROR_NULL_POINTER : _to->GetPartial(aPartial); } \
#define CALL_UPPER_CASE(INP, OUTP, LENP) Perl_to_utf8_case(aTHX_ INP, OUTP, LENP, &PL_utf8_toupper, "ToUc", "utf8::ToSpecUc")
#if (TARG_LOWPART_PLUS_1_U + 61) * (TARG_LOWPART_PLUS_1 << 1) != 61 * (TARG_LOWPART_PLUS_1 << 1) /* { dg-bogus "overflow" } */
else if ((i == 0xA25B) || (i == 0xA25C) || ((i >= 0xB0AE) && (i <= 0xB3C2)) || ((i >= 0xD44B) && (i <= 0xD850))) return 0xB0AE;
{0, { {"simgE", 5} } }, {1, { {(void *)multi_cp_html5_02AA1} } }, {1, { {(void *)multi_cp_html5_02AA2} } }, {0, { {NULL, 0} } },
err = AESendMessage(&event, /*reply*/ NULL, kAENoReply | kAEDontReconnect | kAENeverInteract | kAEDontRecord, kAEDefaultTimeout);
{1, { {(void *)multi_cp_html5_022B4} } }, {1, { {(void *)multi_cp_html5_022B5} } }, {0, { {"origof", 6} } }, {0, { {"imof", 4} } },
NS_SCRIPTABLE NS_IMETHOD GetIsUpgrade(PRBool *aIsUpgrade) { return !_to ? NS_ERROR_NULL_POINTER : _to->GetIsUpgrade(aIsUpgrade); } \
array(null => 1, NULL => 2, "\a" => 3, "\cx" => 4, "\e" => 5, "\f" => 6, "\n" => 7, "\t" => 8, "\xhh" => 9, "\ddd" => 10, "\v" => 11),
if (cs_ctx_alloc(CTLIB_VERSION, &sybase_globals->context)!=CS_SUCCEED || ct_init(sybase_globals->context, CTLIB_VERSION)!=CS_SUCCEED) {
{0, { {"npr", 3} } }, {0, { {"NotSucceeds", 11} } }, {1, { {(void *)multi_cp_html5_02282} } }, {1, { {(void *)multi_cp_html5_02283} } },
if (bundleIdentifier && ([bundleIdentifier compare:GROWL_PREFPANE_BUNDLE_IDENTIFIER options:bundleIDComparisonFlags] == NSOrderedSame)) {
if (_HAS_NONLATIN1_FOLD_CLOSURE_ONLY_FOR_USE_BY_REGCOMP_DOT_C_AND_REGEXEC_DOT_C(value) && (! isASCII(value) || ! MORE_ASCII_RESTRICTED)) {
if (flags < 0 || flags > (PHP_FILE_USE_INCLUDE_PATH | PHP_FILE_IGNORE_NEW_LINES | PHP_FILE_SKIP_EMPTY_LINES | PHP_FILE_NO_DEFAULT_CONTEXT)) {
ERR("Simultaneous use of GL_CONSTANT_ALPHA/GL_ONE_MINUS_CONSTANT_ALPHA and GL_CONSTANT_COLOR/GL_ONE_MINUS_CONSTANT_COLOR invalid under WebGL");
else if ((i == 0xA25E) || (i == 0xF9D7) || (i == 0xF9D9) || ((i >= 0xBEA7) && (i <= 0xC074)) || ((i >= 0xE8F4) && (i <= 0xECB8))) return 0xBEA7;
new Charset("EUC_CN", "GB2312", new String[] {"x-EUC-CN", "csGB2312", "euccn", "euc-cn", "gb2312-80", "gb2312-1980", "CN-GB", "CN-GB-ISOIR165"}),
final int flags = GENERATE_WORD_PARTS | GENERATE_NUMBER_PARTS | CATENATE_ALL | SPLIT_ON_CASE_CHANGE | SPLIT_ON_NUMERICS | STEM_ENGLISH_POSSESSIVE;
{ mbfl_no_language_korean, php_mb_default_identify_list_kr, sizeof(php_mb_default_identify_list_kr) / sizeof(php_mb_default_identify_list_kr[0]) },
final static int ADDRESS_OR_ARTICLE_OR_ASIDE_OR_DETAILS_OR_DIR_OR_FIGCAPTION_OR_FIGURE_OR_FOOTER_OR_HEADER_OR_HGROUP_OR_NAV_OR_SECTION_OR_SUMMARY = 51;
/* { dg-final { scan-assembler "mulhu\tr(\[0-9]\|\[1-2]\[0-9]\|3\[0-1]),r(\[0-9]\|\[1-2]\[0-9]\|3\[0-1]),r(\[0-9]\|\[1-2]\[0-9]\|3\[0-1])\[^0-9]" } } */
{0, { {"NotSubsetEqual", 14} } }, {0, { {"NotSupersetEqual", 16} } }, {1, { {(void *)multi_cp_html5_0228A} } }, {1, { {(void *)multi_cp_html5_0228B} } },
```

Data, not code



Software, but not source code



A portion of Windows binary/object code, viewed inside a word processor

1. What is source code?: Definitions

a. Source code is a human-readable form of software
b. 17 USC 101: “A ‘computer program’ is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.”
c. US FDA: “source code. (IEEE) (1) Computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler or other translator. (2) The human readable version of the list of instructions [program] that cause a computer to perform a task.”
d. One might ask: Why does the FDA care about source code?
e. US ITC: example of PO case-specific source-code definitions

Source code definition from US ITC Model PO

[Para. #]. Source Code. A supplier may designate documents, information, or things as "CONFIDENTIAL SOURCE CODE—ATTORNEY'S EYES ONLY INFORMATION," which shall mean Litigation Material of a supplier or of any non-parties that a supplier is permitted to produce in this Investigation that constitutes or contains non-public Source Code.

A. "Source Code" shall mean source code and object code (*i.e.*, computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler, or other translator). For avoidance of doubt, this includes source files, make files, intermediate output files, executable files, header files, resource files, library files, module definition files, map files, object files, linker files, browse info files, and debug files.

B. Materials designated as "CONFIDENTIAL SOURCE CODE—ATTORNEY'S EYES ONLY INFORMATION," shall only be reviewable by SOURCE CODE QUALIFIED PERSONS. SOURCE CODE QUALIFIED PERSONS include the following: (1) Outside Litigation Counsel as necessarily incident to the litigation of this Investigation; (2) personnel at

Source code definition from a PO

IT IS HEREBY ORDERED THAT Order No. 1, the Protective Order for this investigation, is supplemented with the following provisions:

24. Documents designated "[supplier's name] CONFIDENTIAL BUSINESS INFORMATION-SOURCE CODE, SUBJECT TO PROTECTIVE ORDER" shall be provided with the following further protections:

A. Source Code includes human-readable programming language text that defines software, firmware, or electronic hardware descriptions (hereinafter referred to as "source code"). Text files containing source code shall hereinafter be referred to as "source code files." Source code files include, but are not limited to files containing source code written in "C", "C++", assembler, VHDL, Verilog, and digital signal processor (DSP) programming languages. Source code files further include ".include files," "make" files, link files, and other human-readable text files used in the generation and/or building of software directly executed on a microprocessor, micro-controller, or DSP. Source code does not include binary executable files and object code files, nor does it include tools such as compilers or linkers.[1]

[1] The parties agree that binary executable files and object code files do not need to be produced. To the extent

1. What is source code?: Software analogies

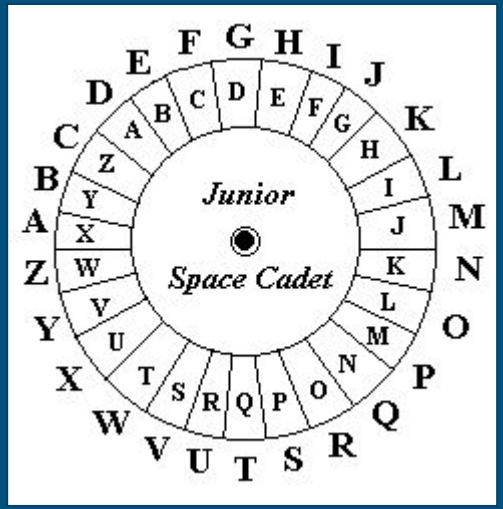
a. Some imperfect analogies for software
b. “Blueprints” -- how same/diff from source code
c. Recipes -- e.g. “The Sachertorte Algorithm”
d. “Piano roll blues”
e. Munitions or speech?: DeCSS (on t-shirts, ties), but Stuxnet

```
s''$/=\2048;while(<>){G=29;R=142;if((@a=unqT="C*",_)[20]&48){D=89;_=
b=map{ord qB8,unqb8,qT,_^$a[--D]}@INC;s/...$/1$&/;Q=unqV,qb25,_;H=73
|256|$b[3];Q=Q>>8^(P=(E=255)&(Q>>12^Q>>4^Q/8^Q))<<17,O=O>>8^(E&(F=(S=O>>14&7^O)
^S*8^S<<6))<<9,_=(map{U=_%16orE^=R^=110&(S=(unqT,"\xb\ntd\xbz\x14d")[_/16%8]);E
^=(72,@z=(64,72,G^=12*(U-2?0:S&17)),H^=_%64?12:0,@z)[_%8]}(16..271))[_]^((D>>=8
)+=P+(~F&E))for@a[128..$#a]}print+qT,@a}';s/[D-HO-U_]/\$$&/g;s/q/pack+/g;eval
```

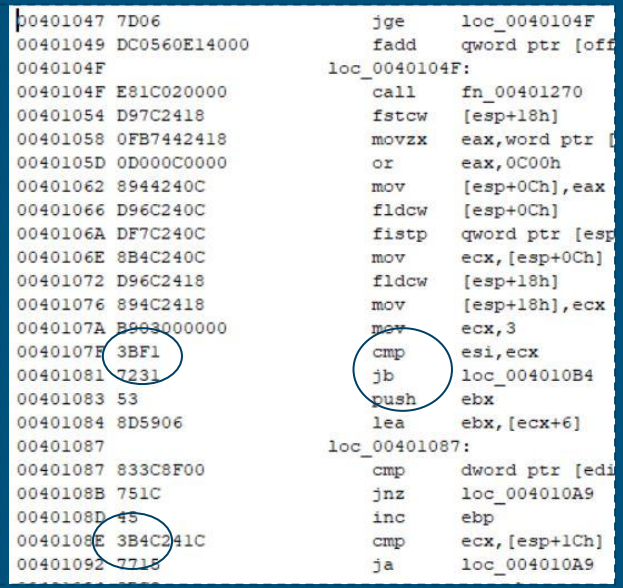
1. What is source code?: Why software is called “code”

a. NOT “code” in the sense of something secret
b. Capt. Marvel Decoder Ring
c. Attorney in discovery dispute: “They have to give us all their Codes”
d. Code here = numbers that machine can interpret as instructions
e. Instructions -- SHOW machine code & ASM, from prime # program
f. Code as a special (operable) kind of text or document
g. Source code often indirectly operable

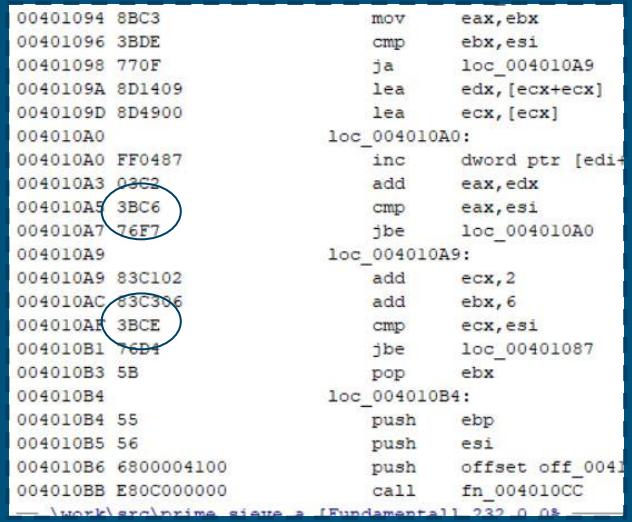
“They have to give us Their Codes”



Code = instructions to a machine



```
00401047 7D06                    jge     loc_0040104F
00401049 DC0560E14000            fadd    qword ptr [off
0040104F                     loc_0040104F:
0040104F E81C020000              call    fn_00401270
00401054 D97C2418                fstcw   [esp+18h]
00401058 0FB7442418              movzx   eax,word ptr [
0040105D 0D000C0000              or      eax,0C00h
00401062 8944240C                mov     [esp+0Ch],eax
00401066 D96C240C                fldcw   [esp+0Ch]
0040106A DF7C240C                fistp   qword ptr [esp
0040106E 8B4C240C                mov     ecx,[esp+0Ch]
00401072 D96C2418                fldcw   [esp+18h]
00401076 894C2418                mov     [esp+18h],ecx
0040107A B903000000              mov     ecx,3
0040107F 3BF1                    cmp     esi,ecx
00401081 7231                    jb      loc_004010B4
00401083 53                      push    ebx
00401084 8D5906                  lea     ebx,[ecx+6]
00401087                     loc_00401087:
00401087 833C8F00                cmp     dword ptr [edi
0040108B 751C                    jnz     loc_004010A9
0040108D 45                      inc     ebp
0040108E 3B4C241C                cmp     ecx,[esp+1Ch]
00401092 7715                    ja      loc_004010A9
```



```
00401094 8BC3                    mov     eax,ebx
00401096 3BDE                    cmp     ebx,esi
00401098 770F                    ja      loc_004010A9
0040109A 8D1409                  lea     edx,[ecx+ecx]
0040109D 8D4900                  lea     ecx,[ecx]
004010A0                     loc_004010A0:
004010A0 FF0487                  inc     dword ptr [edi+
004010A3 03C2                    add     eax,edx
004010A5 3BC6                    cmp     eax,esi
004010A7 76F7                    jbe     loc_004010A0
004010A9                     loc_004010A9:
004010A9 83C102                  add     ecx,2
004010AC 83C306                  add     ebx,6
004010AF 3BCE                    cmp     ecx,esi
004010B1 76D4                    jbe     loc_00401087
004010B3 5B                      pop     ebx
004010B4                     loc_004010B4:
004010B4 55                      push    ebp
004010B5 56                      push    esi
004010B6 6800004100              push    offset off_0041
004010BB E80C000000              call    fn_004010CC
```

For example, 3B corresponds to a CMP (compare) instruction

1. What is source code?: “The Crown Jewels”

a. “Our source code is our Crown Jewels” …
b. … which often the owner can’t quite identify or put their hands on when it comes time to produce them in discovery
c. In damages, the crown jewels may suddenly become “oh that old thing”
d. Source code may embody not only an organization’s IP, but also its policies, “business rules,” and practices

Policy / Business Rules embodied in code

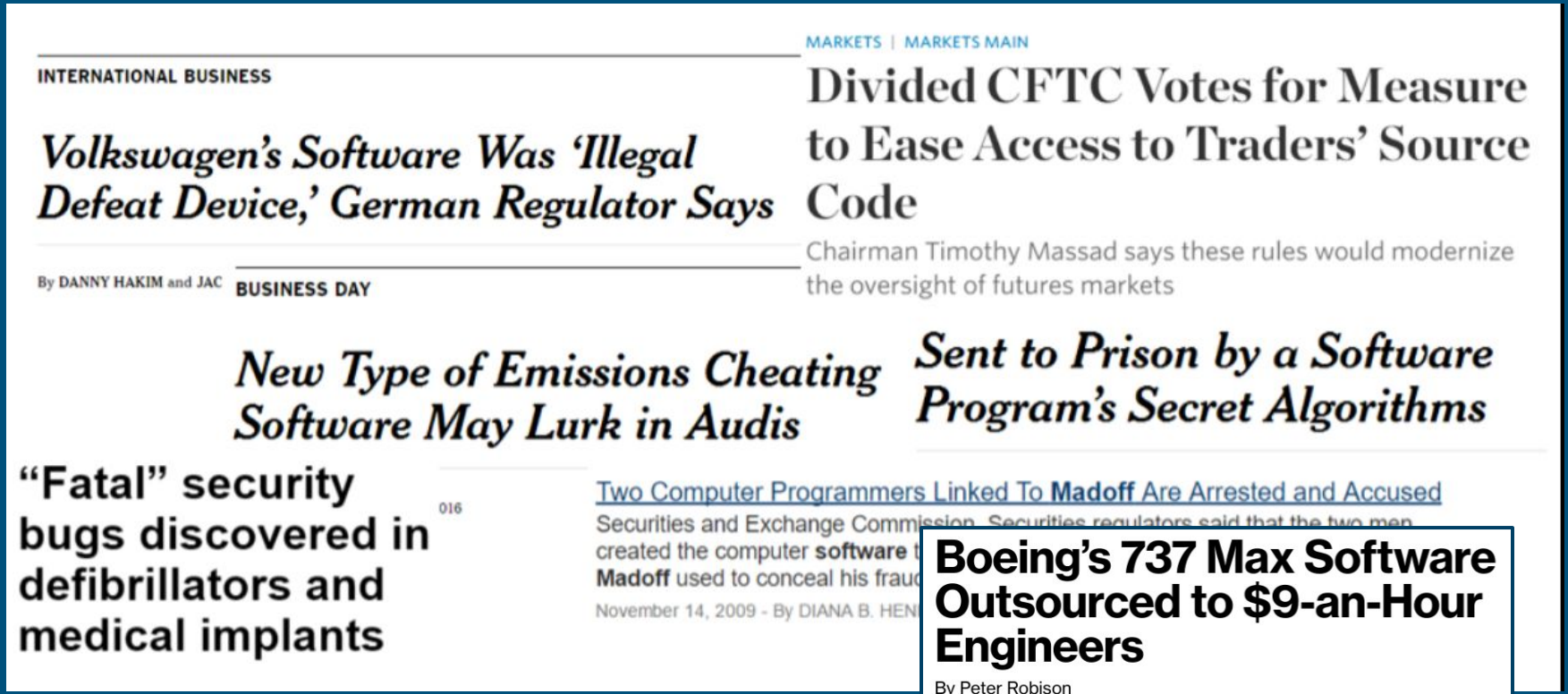
```
// keep this employee's logs for more than 30 days?
boolean preserveUserLogs (User user) {
    boolean doPreserveLogs = false;
    if (user.getHits() > MIN_HITS_TO_ENABLE_PRESERVE)
       doPreserveLogs = true;

    if (doPreserveLogs == true) {
        user.setPreserveState(true);
        user.setPreserveDays(user.getPreserveDays() + PRESERVE_DAYS_INCR); // TODO: check/fix rollover
    }

    return doPreserveLogs;
}
```

```
boolean initializeSystem() {
    // ...
    customerIniFile = getCustIniFile(); // default name: userRecorder.ini
    // ...
    MIN_HITS_TO_ENABLE_PRESERVE = customerIniFile.getMinHits();
    if (MIN_HITS_TO_ENABLE_PRESERVE < 10) // customer not change default, or too low
        MIN_HITS_TO_ENABLE_PRESERVE = 10; // default 10
    PRESERVE_DAYS_INCR = customerIniFile.getpreserveDays();
    if (PRESERVE_DAYS_INCR < 5) // customer not change default, or too low
        PRESERVE_DAYS_INCR = 5; // default 5
    // ...
}
```

2. Why should lawyers care? How can you use source code in litigation?

a. Headline-worthy software
b. Software is everywhere
c. Practice areas
d. Specific types of questions source code can answer
e. Not only in software cases: e.g. forensics, policy, models

Newsworthy software



2. Who cares?: “Software is everywhere”: my practice

- Electric arc steel furnaces (industrial control)
- Limousine services
- Metal detectors
- Actor motion/gesture capture
- HIV dating
- Machine tools, robotics
- Hedge fund
- DNA microarray
- Automotive firmware
- Door locks (Bluetooth, IoT; mobile phone as intelligent key)
- Chinese online payment systems
- Mobile check deposit
- Cable modems
- Video teleconferencing
- Online shopping & advertising
- Web Rx ordering
- … as well as software as such (Microsoft Windows, Microsoft Office, Apple iOS, mobile apps)

“Software is everywhere”: DisputeSoft cases with source-code examination

- Hospital dispute with medical software vendor (support K)
- State health insurance exchange (pre-litigation investigation)
- Smartphone touch screen patents
- Prison telephone system patent
- Property management (K breach)
- Casualty policy admin & billing (arbitration re: COBOL to Java)
- Aviation maintenance ©
- County property tax management (K software failure)
- Payroll software ©
- Oil & gas exploration ©
- Tenant management system (K project failure)
- Exam prep materials © & TS
- Online sales leads ©
- Social media TS (Facebook)
- Source audit due diligence in M&A

2. Who cares?: Source code in different legal practice areas

- IP: ©, patent, TS (not TM), including network, software-based devices
- Antitrust: e.g. Microsoft tying, expensive monopolization acts
- Torts / Products liability: medical devices, auto, aviation, industrial
- Contracts: project failure, defects
- Criminal law: code theft, code fraud (emissions), questioning forensics devices/output (Confrontation Clause), sentencing predictions
- Employment law: TS = IP + employment law; company policy -- SHOW
- Environment law: EPA models (but careful with “statistical populism”)

2. Why cares?: Legal practice areas, continued

- Constitutional law: voting machines; code as speech; public library internet filtering; evidence of bias; Confrontation Clause & forensic devices (*Bullcoming v. NM* on machine-generated evidence)
- Regulatory compliance: HIPAA, FISMA, SOX, GDPR, CFTC
- Tax law: software depreciation, amortization
- Privacy: HIPAA; GDPR; class actions; employer monitoring -- SHOW
- Mergers/acquisitions (M&A) due diligence: open source audits
- Cybersecurity: corporate liability for data breach by hacker

2. Who cares?:
Some specific questions source code can answer

- Does this product, service, or in-house process do/contain?; equivalent?
- Are there similarities (literal or not) between these products due to copying?
- Is there an error or unusual feature common to both products?
- Was this code protected by reasonable security precautions?
  (RSP for TS; cybersecurity liability after data breach)
- Does the company's manual match its actual *de facto* policy?
- Who wrote this code?
- Did this internal email come to fruition?
- Is this bug within reasonable industry standards? Fit for intended use?
- Is this software's output real or fake?
- Is this computer-generated forensics output sufficiently reliable?

2. Who cares?:
More specific questions source code can answer

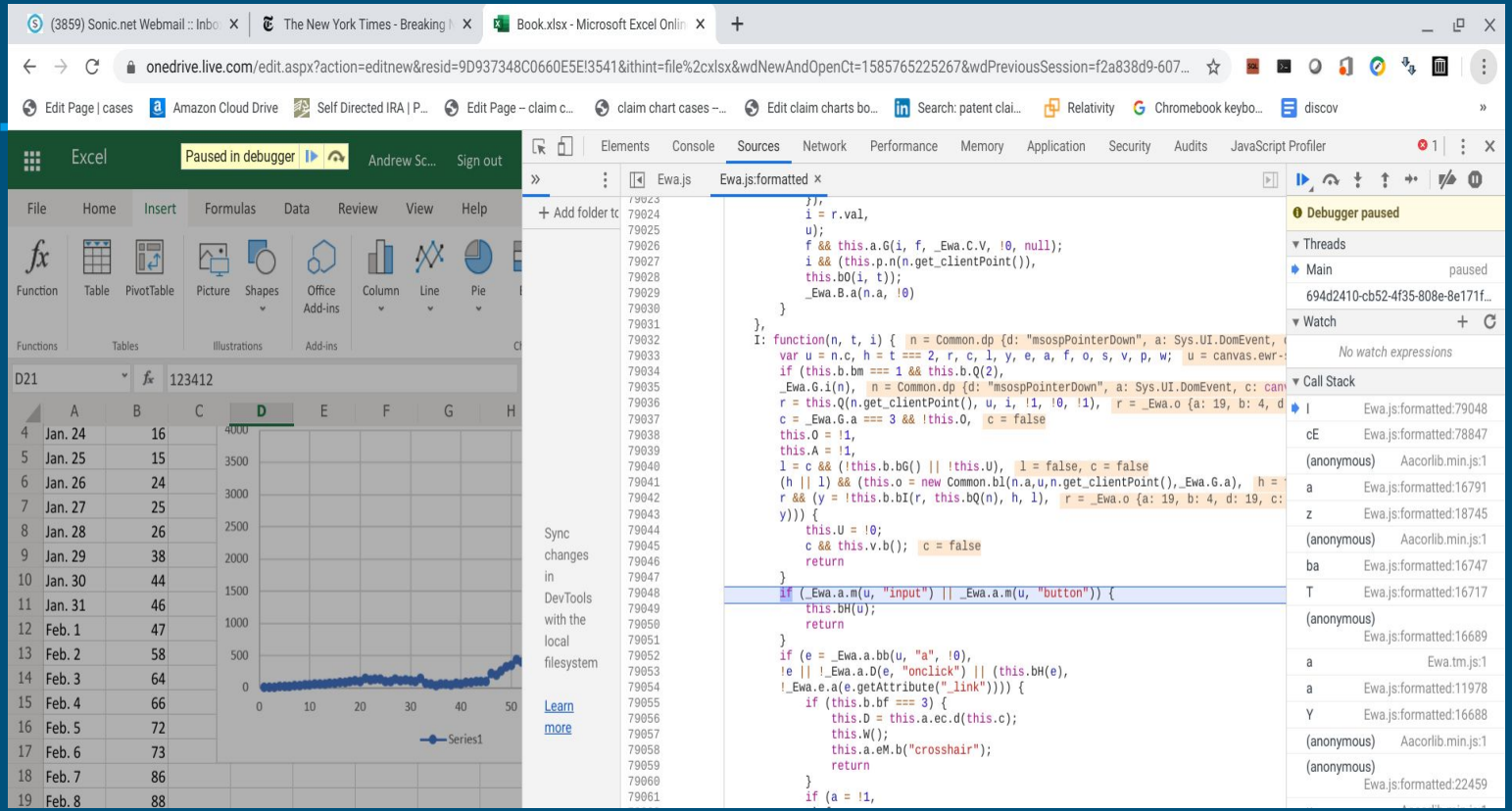
- What % overlap is there between these two pieces of software?
- Where did this computer-generated evidence (CGE) come from?
- What statistical model are these projections based upon?
- What assumptions does this device or process make?
- What bugs or errors does this contain, which could affect results?
- What are this organization's business rules?
- Were there good contemporaneous reasons to not fix this bug?
- What risks or vulnerabilities might this code present? (Software Composition Analysis (SCA) for e.g. M&A due diligence)

What source can't CAN'T answer: static vs. dynamic views of software (see later discussion of reverse engineering)

3. Types of source code; important distinctions

a. Binary/object code vs. source code vs. open/readable proprietary code; SHOW JavaScript code for Excel Online in Chrome Web Developer
b. High-level language (HLL) vs. assembly language (ASM) -- SHOW
c. Programming languages: C++, Java, JavaScript (JS), Perl, Python, SQL, R…
d. Interpreted vs. compiled; scripts, batch/cmd files
e. Standalone vs. client/server, network, mobile
f. In-house software vs. product/server on the market
g. Firmware, embedded software
h. Platforms: Android, iOS, OSX, Windows, Linux; mobile vs. desktop
i. Application vs. library: APIs, SDKs

Microsoft Excel Online, JS in Web Developer



Machine language, assembly, and C

```
00401047 7D06                jge     loc_0040104F
00401049 DC0560E14000        fadd    qword ptr [off_0040E160]
0040104F                     loc_0040104F:                   ; Xref 00401047
0040104F E81C020000              call    fn_00401270
00401054 D97C2418                fstcw   [esp+18h]
00401058 0FB7442418              movzx   eax,word ptr [esp+18h]
0040105D 0D000C0000              or      eax,0C00h
00401062 8944240C                mov     [esp+0Ch],eax
00401066 D96C240C                fldcw   [esp+0Ch]
0040106A DF7C240C                fistp   qword ptr [esp+0Ch]
0040106E 8B4C240C                mov     ecx,[esp+0Ch]
00401072 D96C2418                fldcw   [esp+18h]
00401076 894C2418                mov     [esp+18h],ecx
0040107A B903000000              mov     ecx,3
0040107F 3BF1                    cmp     esi,ecx
00401081 7231                    jb      loc_004010B4
00401083 53                      push    ebx
00401084 8D5906                  lea     ebx,[ecx+6]
00401087                     loc_00401087:                   ; Xref 004010B1
00401087 833C8F00                cmp     dword ptr [edi+ecx*4],0
0040108B 751C                    jnz     loc_004010A9
0040108D 45                      inc     ebp
0040108E 3B4C241C                cmp     ecx,[esp+1Ch]
00401092 7715                    ja      loc_004010A9
00401094 8BC3                    mov     eax,ebx
00401096 3BDE                    cmp     ebx,esi
00401098 770F                    ja      loc_004010A9
0040109A 8D1409                  lea     edx,[ecx+ecx]
0040109D 8D4900                  lea     ecx,[ecx]
004010A0                     loc_004010A0:                   ; Xref 004010A7
004010A0 FF0487                  inc     dword ptr [edi+eax*4]
004010A3 03C2                    add     eax,edx
004010A5 3BC6                    cmp     eax,esi
004010A7 76F7                    jbe     loc_004010A0
004010A9                     loc_004010A9:                   ; Xref 0040108B 00401092 004
004010A9 83C102                  add     ecx,2
004010AC 83C306                  add     ebx,6
004010AF 3BCE                    cmp     ecx,esi
004010B1 76D4                    jbe     loc_00401087
004010B3 5B                      pop     ebx
004010B4                     loc_004010B4:                   ; Xref 00401081
004010B4 55                      push    ebp
004010B5 56                      push    esi
004010B6 6800004100              push    offset off_00410000     ; '%lu => %lu',00Ah,
004010BB E80C000000              call    fn_004010CC
```

```
main(int argc, char *argv[])
{
        INT max = (argc < 2) ? 1000 : atol(argv[1]);
        INT *arr = (INT *) calloc(max+1, sizeof(INT));
        INT cnt = 1;
        INT sqrt_max = (INT) sqrt(max);
        INT i, j;

        for (i=3; i<=max; i+=2) {
                if (arr[i] == 0) {
                        cnt++;
                        if (i <= sqrt_max) {
                                for (j=i+i+i; j<=max; j+=(i+i))
                                        arr[j]++;
                        }
                }
        }

        printf("%lu => %lu\n", max, cnt);
        // e.g. 1000 => 168
        // e.g. 100000000 => 5761455 (first 5.7 million primes)
}
```

3. Types of source code -- continued

j. Database code: SQL, including stored procedures
k. Auto-generated code, “wizards”
l. Statistical code: SAS, R
m. Excel: cell formulas; VBA scripts

Where are we?

1. What is source code?
2. Why should attorneys care? What can you do with it in litigation?
3. Types of source code, and some important distinctions
4. Timeline of source-code use in litigation
5. Discovery, protective orders (POs)
6. Experts & source-code examination skills & methodology
7. How source code relates to computer forensics, e-discovery, etc.
8. Some cases
9. Trends & take-aways

4. A timeline of source-code use in litigation

- **Request for production** or for inspection (or mandatory disclosure obligation e.g. under LPRs)
- Attorney has client custodians **inventory** its ESI, including source code
- **Produce under PO** to other side's experts, generally in environment with constraints; maybe under 33(d) rog response
- **Exam**: Initial inventory of what was produced (sometimes massive)
- Indexing, possibly with regularization, cleaning
- Initial search with initial keywords
- Refining/expanding keywords, reading, tracing, note-taking (see PO)

4. Timeline, continued

- **Select, extract, and/or print** few files
- **Review** by producing party, then **Bates** stamp, send to requesting party
- Reading/analyze (except some POs say only *in situ* during exam)
- Possibly repeat search/extract/review/analyze
- Possibly repeat requests, **repeat visits** for missing or new code
- **Expert report**, affidavit (PO issues -- AEO)
- Expert deposition -- before or after see other side's report?

4. Timeline, continued

- Repeat?: other side's report, rebuttal report may inspire another visit
- **Amend or supplement** pleadings, if good cause / diligence
- **Securely maintain** source code or printouts per PO
- Possibly demonstrative exhibits, though usually not "smoking gun"
- Trial testimony
- **Return or destroy** code and code-based materials

5. Discovery and Protective Orders (POs)

a. Discovery: production & requests
b. Who, what, when, where, why, how, how much
c. Protective orders (POs)
d. Third-party code: commingling & subpoenas
e. Missing code, alterations & spoliation
f. Gamesmanship & proportionality; know why you are asking for (or refusing) source code -- specific narrowly-tailored requests more likely to lead to source code discoverability

5. Source code discovery: a magistrate's view

"In a typical patent infringement case involving computer software, few tasks excite a defendant less than a requirement that it produce source code. Engineers and management howl at the notion of providing strangers, and especially a fierce competitor, access to the crown jewels. Counsel struggle to understand even exactly what code exists and exactly how it can be made available for reasonable inspection. All sorts of questions are immediately posed:

- Exactly who representing the plaintiff gets access—and does this list include patent prosecution counsel, undisclosed experts, and so-called 'competitive decision makers'?
- Must requirements and specification documents that explain the functionality implemented by the code be included?
- What compilation, debugging and analysis tools are required?
- ...

5. Discovery: Apple v. Samsung, continued

- What about the test database and user manuals?
- Make files? Build files?
- Does the code have to [be] produce[d] in a native repository such as CVS or Perforce?
- Must daily builds in development be produced (and if so, in real-time or batch?) or is production limited only to copies in commercial release?

Put simply, source code production is disruptive, expensive, and fraught with monumental opportunities to screw up."

Apple v. Samsung (ND Cal 2012), ORDER GRANTING APPLE'S MOTION FOR 37(B)(2) SANCTIONS RE DECEMBER 22 DISCOVERY ORDER

5. Source code discovery: who, what, when, ...

- WHO gets to see source code? (PO, AEO)
- WHOSE source code: D's, P's, third party (3P)?
- WHAT/WHICH source code: specific versions, forthcoming, old?
- WHAT types of files: build scripts, logs, object code, etc.? (how "source code" is defined in this case's PO)
- WHEN is the source code from: old from backup, forthcoming?
- WHEN is the source code produced: rolling production?
- WHERE is the source code located in the organization: central repository, dispersed, employee homes, on backups, in the cloud?

5. Source code discovery: who, what, when, ...

- WHERE will the source be produced to the other side:
  sent to expert, in cloud, on-site, law firm, escrow facility?
- WHY are you asking for source code? (know why you are asking,
  what questions you expect to have answered; fishing expeditions)
- WHY are you refusing? (really TS concern, or want to stick it to
  the other side e.g. because “troll” without MAD?)
- HOW: how will the source code be produced?; format: native,
  inside version control, original path/filenames, metadata
- HOW MUCH: dump all versions (careful what you ask for), sample?

5. Source code discovery & POs, continued

- Protective orders (POs)
- Third-party (3P) code: drop-in libraries; possession/custody/control; commingling; 3P subpoenas; discovery rules enforced tightly for 3P
- Missing code, altered code, “redactions” & spoliation
- Proportionality; know why you are asking for (or refusing) source code
- Amending & supplementing, good cause vs. “reserve right”, diligence
- Generally few authentication, privilege, admissibility issues

5. Source code protective orders (POs)

- Blanket/umbrella designation of ALL source code as TS or CBI, even though most contains large amounts of public/open source
- AEO for CBI; outside counsel; patent prosecution bar
- PO largely defines source-code examination environment: standalone computer, no USB, no internet -- so no comparisons
- Tools placed on standalone exam computer
- Printing limits
- Some POs require all analysis on site
- Will there be a trend towards remote/cloud source-code access?
- See article on impact of POs on source code exam; AJL article

5. Missing/altered code & spoliation

- Source code examiner might overlook
- Not all source code centralized in version control: scripts; Microsoft
- Older code on less-accessible backups
- Custodian inventory often seems half-hearted; “policy” vs. reality
- Producing client code, but not server code
- Producing Windows code, but not Android, iPhone/iOS, Mac/OSX
- Not producing with original folder/file names
- “Redacting” code (especially comments)
- Many spoliation cases: destroying, or allowing alteration
- Losing the “crown jewels,” or turns out Crown doesn’t know its jewels

6. Experts: source-code exam skills, methods

a. Background/skills needed for source-code exam in litigation
b. Tools used in source-code exam
c. Source code exam method/process: before, during, after; Daubert
d. Some source-code problems or “gotchas”
e. Expert report
f. Expert’s own code

6. Experts: background, skills

- Background/skills needed for source-code exam in litigation
- Why not an e-discovery wizard -- “pattern matching”
- Why not just any programmer -- answering litigation questions
- Computer science (CS) or Software engineering (SE)?; networking
- Consulting non-testifying experts; coordination
- Some specific skills...

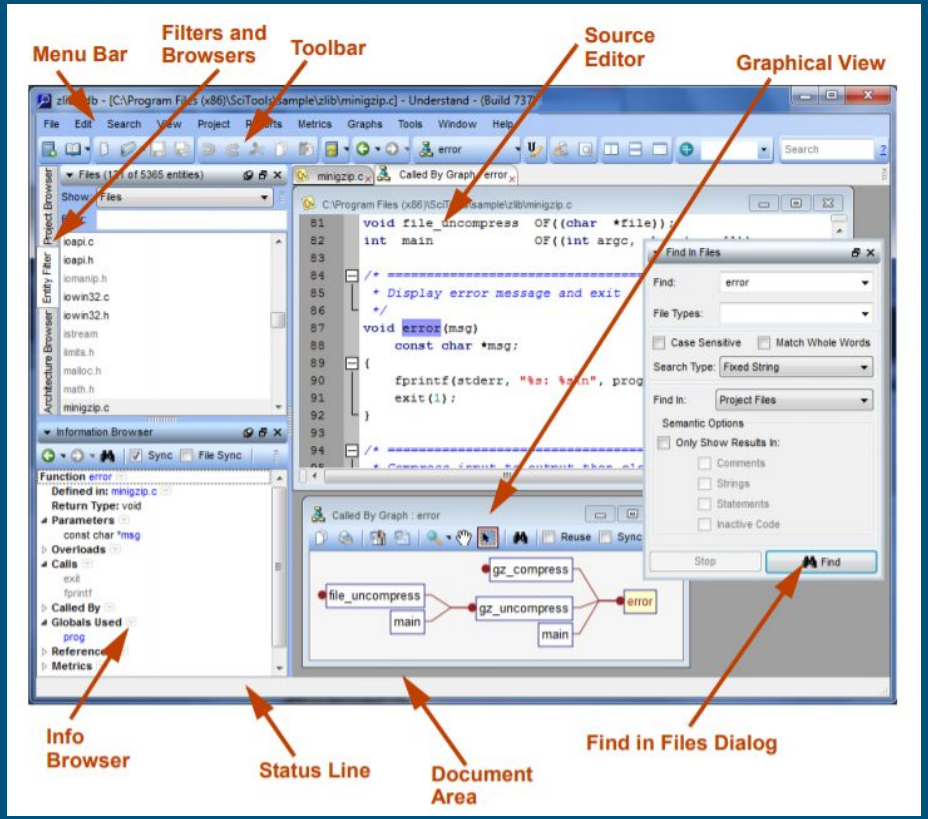
6. Experts: some specific skills; ability to...

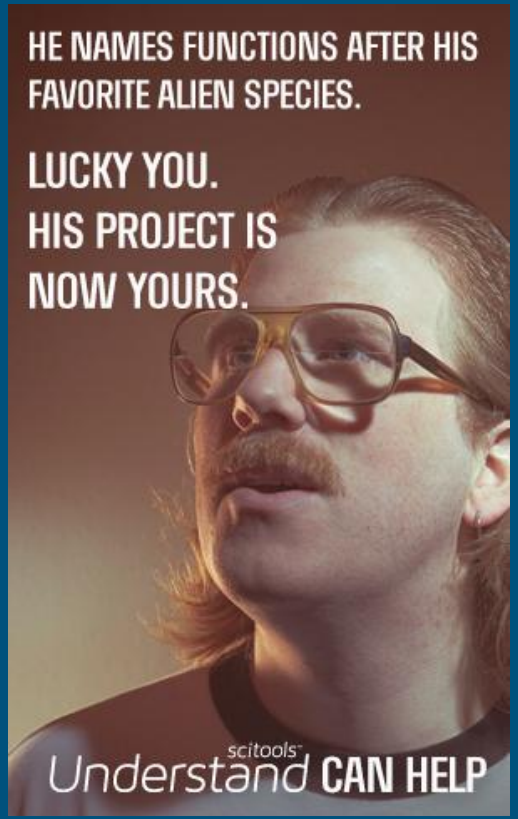
- match X and Y, based on attributes (e.g., F/W/R, I/O); synonyms
- do tracing to/from search hits; not just “pattern matching”
- recognize key “idioms” in code, even if not labelled as such (names or “symbols” in code are just linkages)
- recognize important low-level from high-level, vice versa (e.g. see associative array, know likely hash table underneath)
- recognize constructs when unnamed (anonymous functions), or oddly named (programmer’s favorite alien species, “chilluns”)
- detect when something is missing (not just missing code, but also some necessary element, e.g. of patent claim, is possibly not done)
- test assertions (Wason card problem)

6. Experts: Tools used in source-code exam

- SHOW SciTools Understand (e.g. call tree ~ Shepardizing)
- SHOW WinMerge
- SHOW dtSearch
- SHOW Command line tools: grep, diff, findstr, strings; Cygwin
- SHOW Scripting languages available on locked-down source-code computer under PO: awk, VB (PowerShell)
- Others, often listed in POs; XCode, MSFT Visual Studio, Sigasi (VHDL), PowerGrep, Notepad++, SlickEdit, Eclipse, etc.; EnCase; CodeSuite
- Ad hoc v. off-the-shelf; in-house (CodeExaminer); Daubert issues

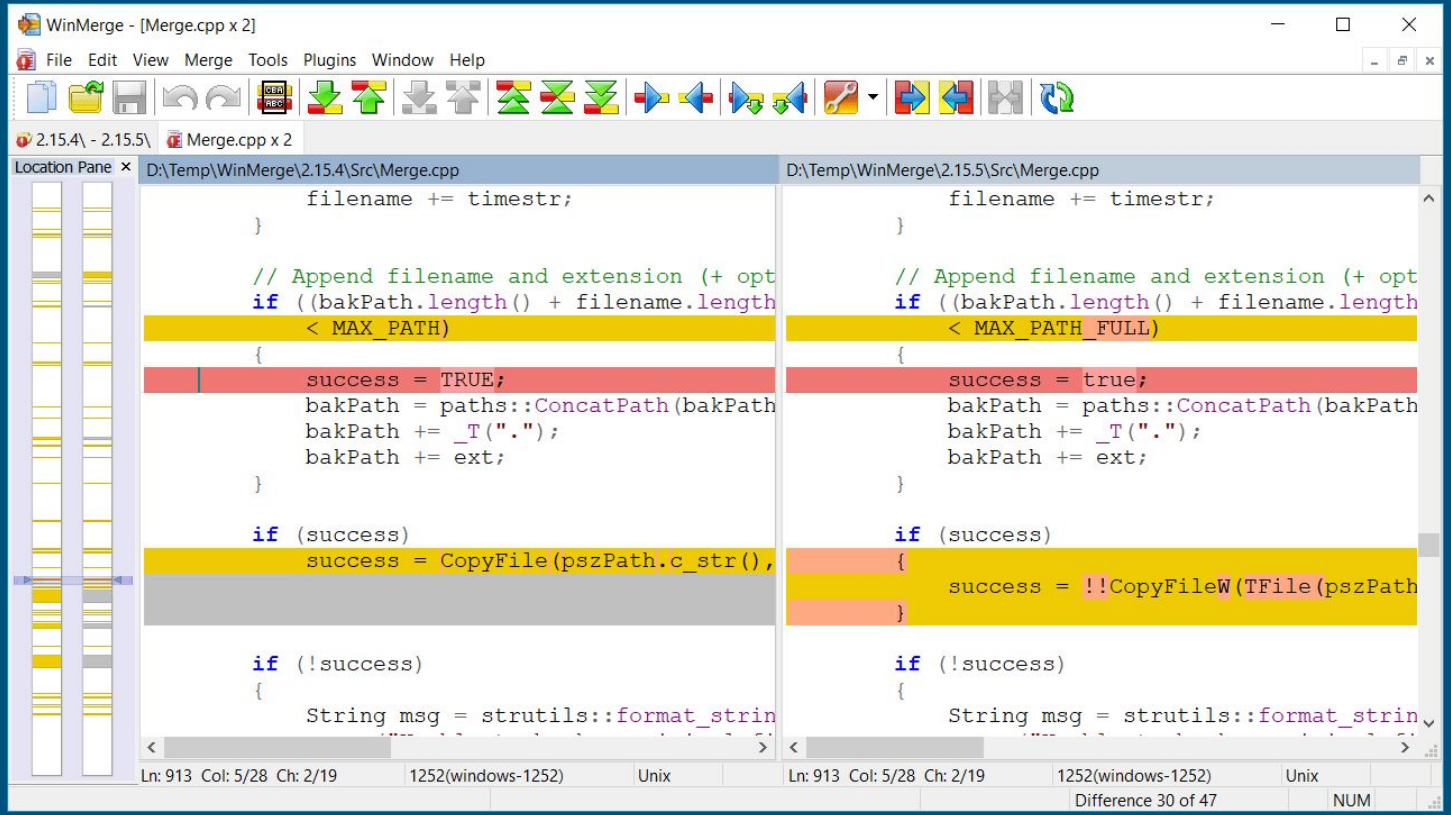
Tools: Understand (SciTools)



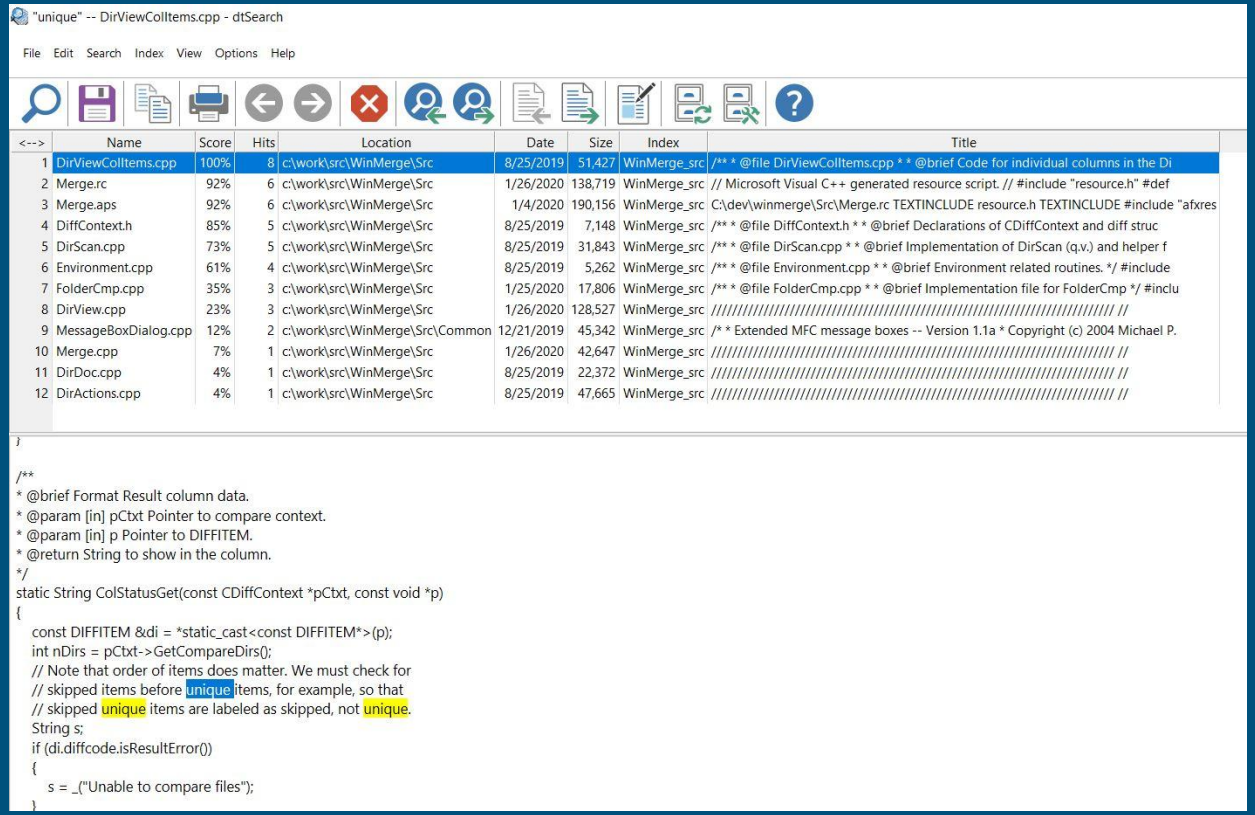


Schulman / Source Code / SoftwareLitigationConsulting.com

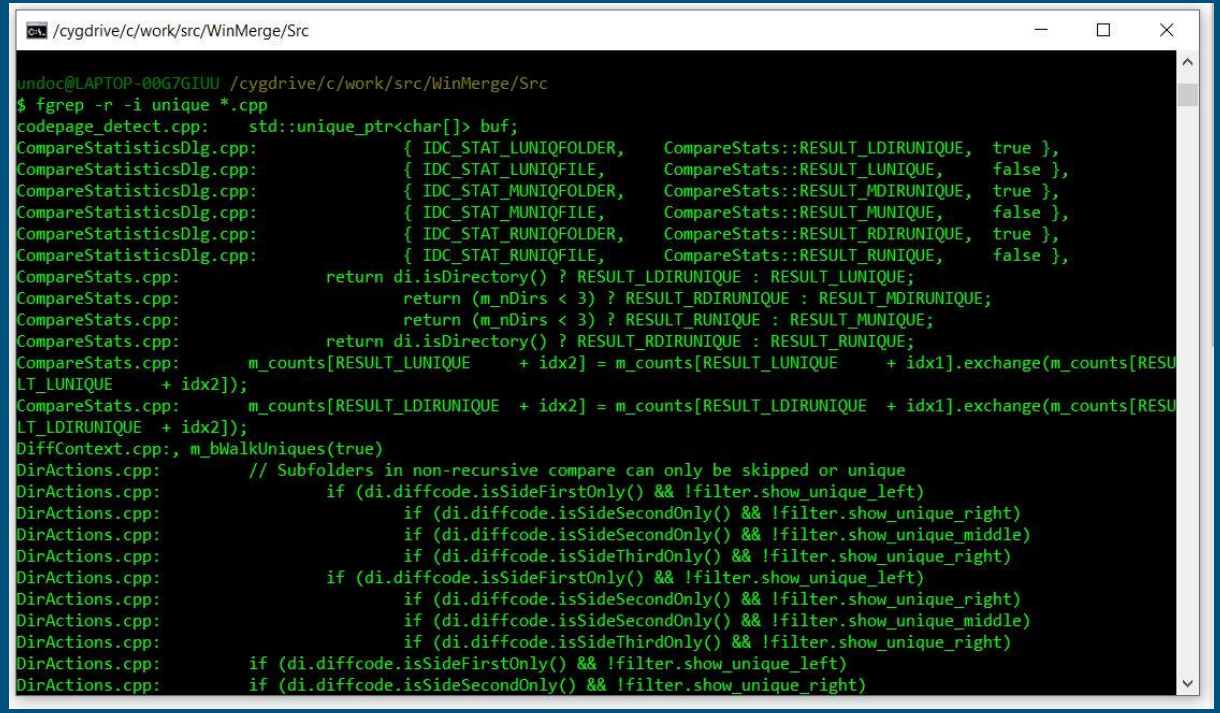
Tools: WinMerge (diff)



Tools: dtSearch



Command-line tools example: grep (regex)



Command-line tools example: awk script

```
C:\work\src>type find_copyright.awk
/Copyright / { # require initial upper-case, space
        gsub(/[ \t]+/, " ", $0);
        sub(/^[ \\\*\/]+/, "", $0);
        arr[$0]++;
}

END {
        for (x in arr)
                print arr[x], "\t", x ;
}


C:\work\src>awkw -f find_copyright.awk WinMerge\Src\*.cpp  | \bin\sort -rn
29      Copyright (C) 1997-2000 Thingamahoochie Software
2       Copyright 1997 Chris Losinger
2       Copyright (c) 2005 Jochen Tucht
2       Copyright (C) 1997 Dean P. Grimm
1       Copyright (c) 2003 Jochen Tucht
1       Copyright (C) 2000 - Francis Irving
1       Copyright (C) 1988, 89, 91, 92, 93 Free Software Foundation, Inc.
```

6. Experts: Method, process

One approach: before/during/after exam

- Before the exam (including pre-filing investigation)
  - If consultant/expert brought on early, help draft discovery requests
  - Diligently mine public info, including reverse engineering public product/service
  - Often can “map” source-code hierarchy from public product
  - Study platform, APIs
  - Establish initial set of specific narrow technical questions
  - Carefully read the PO!
- ....

6. Experts: Method, process

- During the exam
  - Take inventory of source: list directories; count file extensions; list ©, open source, 3P
  - Look for files or keywords known from pre-filing/pre-discovery investigations
  - Pay attention to PO restrictions on note-taking, printing, analysis only on-site
  - Code indexing, searching, reading, tracing, analyzing, comparing (if possible)
  - Follow standard methods, e.g. Spinellis, Code Reading; Microsoft, Code Complete
  - Try to answer specific narrowest, clearest questions first
  - Look for absences: missing code, required negatives (e.g. patent claim elements)
- After the exam
  - Analysis, if allowed by PO
  - Report drafting, possibly paraphrasing source code
  - Possibly amend/supplement, rebut, additional discovery requests, revisit source

6. Experts: source-code exam problems

- Wrong source code produced, possibly because didn't request properly ("You didn't say 'Simon Says'"); produce client without server
- Wrong version of code
- Missing files, missing 3P libraries, owner doesn't have source (!)
- Code produced, but examiner overlooks (odd file extensions, archives)
- Examiner fails to follow PO: printing too much, note-taking
- Producing party fails to follow PO: forgot to turn off USB, internet
- Producing party turns off necessary tools on source-code machine
- PO inhibits analysis necessary for case: e.g. © code comparison

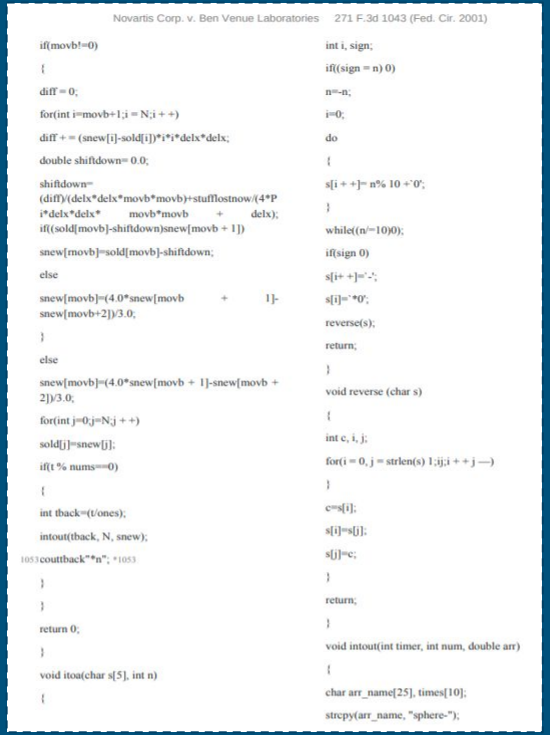
6. Experts: source-code exam “gotchas”

- Jumping from comparisons to conclusions, without baseline
- Assuming names or comments are accurate, or misinterpreting
- Focusing on source code not used in product, or not executed
- Not correlating static source code with dynamic view of public product
- Code path modified at run-time: function pointers, hooks, callbacks
- Unclear entry points into code: public URL -> code that handles
- Implicit or “invisible” code executed: e.g. C++ constructors, destructors
- Missing aliases for names: functions; data flow
- Failure to look for absences, negatives, counter-examples

6. Expert report

- FRCP requirements
- Daubert basis for opinion: path from facts to opinions
- Negative conclusions or absences especially need method description
- PO possibly forbids quoting from code: paraphrasing
- Report may be AEO, redacted for retaining party?
- Rebuttal reports
- Expert's own code -- SHOW Novartis v. Ben Venue Labs

Experts: “inscrutable” code in case opinion



Novartis Corp. v. Ben Venue Laboratories 271 F.3d 1043 (Fed. Cir. 2001)

```
if(movb!=0)
{
diff = 0;
for(int i=movb+1;i = N;i + +)
diff + = (snew[i]-sold[i])*i*i*delx*delx;
double shiftdown= 0.0;
shiftdown=
(diff)/(delx*delx*movb*movb)+stufflostnow/(4*P
i*delx*delx* movb*movb + delx);
if((sold[movb]-shiftdown)snew[movb + 1])
snew[movb]=sold[movb]-shiftdown;
else
snew[movb]=(4.0*snew[movb + 1]-
snew[movb+2])/3.0;
}
else
snew[movb]=(4.0*snew[movb + 1]-snew[movb +
2])/3.0;
for(int j=0;j=N;j + +)
sold[j]=snew[j];
if(t % nums==0)
{
int tback=(t/ones);
intout(tback, N, snew);
1053couttback"*n"; *1053
}
}
return 0;
}
void itoa(char s[5], int n)
{
int i, sign;
if((sign = n) 0)
n=-n;
i=0;
do
{
s[i + +]= n% 10 +'0';
}
while((n/=10)0);
if(sign 0)
s[i+ +]='-';
s[i]='*0';
reverse(s);
return;
}
void reverse (char s)
{
int c, i, j;
for(i = 0, j = strlen(s) 1;ij;i + + j —)
}
c=s[i];
s[i]=s[j];
s[j]=c;
}
return;
}
void intout(int timer, int num, double arr)
{
char arr_name[25], times[10];
strcpy(arr_name, "sphere-");
```

Novartis Corp. v. Ben Venue Laboratories 271 F.3d 1043 (Fed. Cir. 2001)

```
itoa(times, timer);
strcat(arr_name, times);
strcat(arr_name,".dat");
ofstream out1(arr_name);
for(int i=0;i =num;i + +)
{
out1 arr[i]"*n";
}
out1.close();
1054return;
}
```

The program's main routine appears to begin at line 28. The astute reader will have noted that line 28 is not commented. Nor is line 29. Nor is line 30. In fact, this court has searched all the lines of Dr. Nauman's model for comment or explanation, in vain. While we concede that line 12, which defines Pi to be 3.14159, is self-explanatory even to a judge of this court, the remainder of Dr. Nauman's code is populated by inscrutably named variables such as "dollarsmoved" and "centsmoved," which may or may not represent the conditions of Ben Venue's commercial process, upon which are performed unexplained mathematical operations, which may or may not represent the dynamics of Ben Venue's neutralization reaction. Neither the basic theoretical framework nor the derivation of the necessary inputs is apparent from Dr. Nauman's source code.

Dr. Nauman's entire explanation of the basis of his model consists of the following statement:

> I obtained the necessary parameters for the model from the literature (molecular weights of pamidronic acid and pamidronate disodium), from standard correlations (Atkins, *Physical Chemistry*, 5th Ed., p. 24, Freeman, New York, 1987; Perry, *Chemical Engineering Handbook*, 7th Ed., p. 2-372, McGraw Hill, New York, 1997; Sotman, et al., Kristallografiya, Vol. 35, p. 1442, 1990) and from the direct measurements cited above and in the declaration of Prof. McKenna.

*1054 The cited references appear to describe the basic equations for computing reaction rate and diffusivity. They are not specific to any reaction under question and it is unknown how Dr. Nauman employed them. One might estimate the reaction rate for dissolution of pamidronic acid from Dr. McKenna's experiments, but there is no indication how Dr. Nauman derived the rates for the neutralization or crystallization reactions, or if such rates are taken into consideration by the model. Even if we were to accept without question that Dr. Nauman began with the proper parameters, we are left completely in the dark as to how he employed them. It is also unclear if these computations require some assumptions about how the surface zone interacts with the bulk solution-assumptions that might or might not depend on the treatment of the mixing process. Without this information, it is impossible to tell if Dr. Nauman's model accurately reflects the conditions of Ben Venue's process. Apparently, we are to accept as an article of faith that Dr. Nauman employed accepted and realistic equations or theories. Novartis makes some attempt on appeal to explain the input parameters, but even here its explanations are inadequate. Moreover, the theoretical foundation of the model remains inscrutable, and summary judgment does not demand that we refrain from any scrutiny of the nonmovant's evidence.

Where are we?

1. What is source code?
2. Why should attorneys care? What can you do with it in litigation?
3. Types of source code, and some important distinctions
4. Timeline of source-code use in litigation
5. Discovery, protective orders (POs)
6. Experts & source-code examination skills & methodology
7. How source code relates to computer forensics, e-discovery, etc.
8. Some cases
9. Trends & take-aways

7. How source code examination relates to...

a. Computer forensics; code vs. data; except malware RE
b. Forensics generally: individuation vs. classification;
   but examine software used in forensics devices (“black box”)
c. E-discovery, ESI (how source code is similar/different)
d. Other ways of examining software, e.g. reverse engineering
e. Non-litigation source code exam (re: post-Daubert factors)
f. How source code relates to software product on the market

8. Some cases: source-code discovery issues

- Keithley v. HomeStore -- spoliation; allowed source to be destroyed/altered after lawsuit initiated
- OpenTV v. Liberate -- which party bears cost of extracting; Zubulake
- Rosenthal Collins v. Trading Tech -- “turning back the clock” on source code dates, wiping
- MediSim v. BestMed -- testifying expert relying on consultant’s source code exam?
- Unwired Planet v. Apple -- printing limits, quoting source in report
- Advanced Software v. Fiserv -- 4 months source code access ample time to find (or not find) info to amend complaint

8. Sample source code cases

- Apple v. Samsung -- “smartphone wars”; patents, incl. design patents
- US v. Microsoft antitrust, class actions -- tying, anticompetitive acts
- Cisco v. Arista -- ©, TS, ex-employees
- USAA v. WFB -- mobile check deposit patents; 3P Mitek code
- Doe v. PositiveSingles.com -- atty malpractice to not hire src expert?
- State v. Chun (NJ, 2008) -- DUI breathalyzer source code, “black box”
- People v. Johnson (CA, 2018-9) -- TrueAllele DNA forensic software & Confrontation Clause (ACLU)
- Novartis v. Ben Venue Labs (shown earlier)
- REC v. Bamboo (pinpoint source citations in amend patent claim tables)

9. Trends

- Move away from requiring on-site inspection, to producing in cloud?
- Source code using non-English, e.g. Chinese, Korean
- AI, machine learning, models, training
- 3D printing and ©: code or data?
- SaaS, cloud
- Code mining, Big Code, code patterns

Take-aways

a. Think about how you might use source code in a case.
b. Look at computer-generated evidence and ask “How did this get here?”
c. Are an organization’s practices/policies implemented in software? If so, UTSL (“Use the Source, Luke!”)
d. Similarly for a device’s output (machine-generated evidence).
e. But don’t overdo it: “stop and think,” proportionality, alternatives to source code; plausibility & reasonable doubt
f. Source code is another type of doc…
g. … but not just any old doc: special skills needed to read and analyze

For more information...

SoftwareLitigationConsulting.com

DisputeSoft.com

My email: aschulman@disputesoft.com

LinkedIn

Questions?