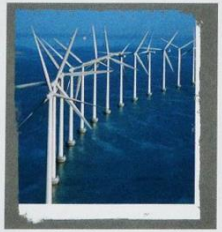

Code as Risk
@KevlinHenney


Kevlin Henney 编
李军 译 吕骏 审校
電子工業出版社
O'REILLY
オライリー・ジャパン
97
Collective Wisdom
from the Experts
97 Things Every
Programmer
Should Know
O'REILLY
Edited by Kevlin Henney

WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

A Pattern Language for Distributed Computing



Volume 4

Frank Buschmann
Kevlin Henney
Douglas C. Schmidt

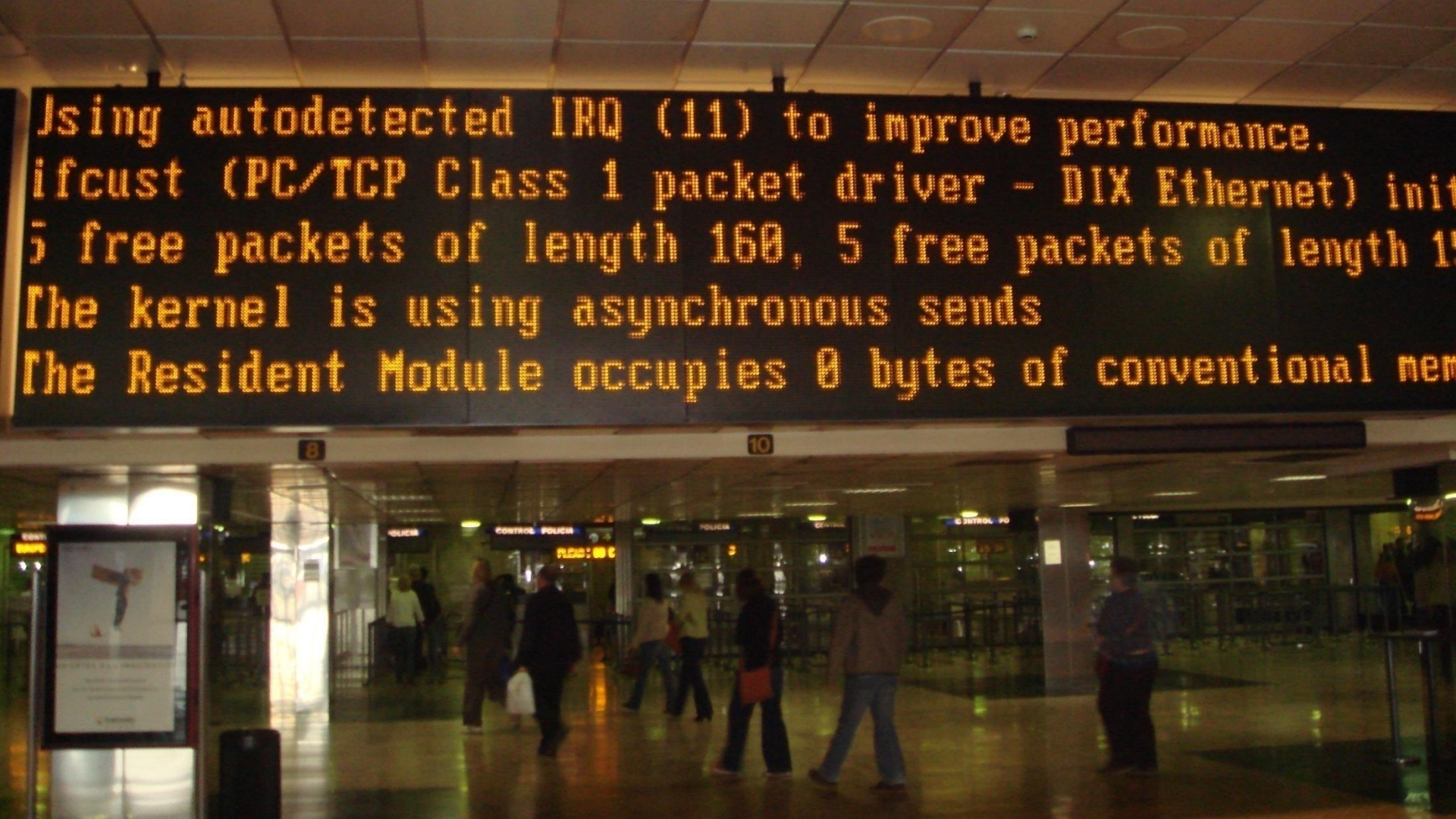
WILEY SERIES IN
SOFTWARE DESIGN PATTERNS

PATTERN-ORIENTED SOFTWARE ARCHITECTURE

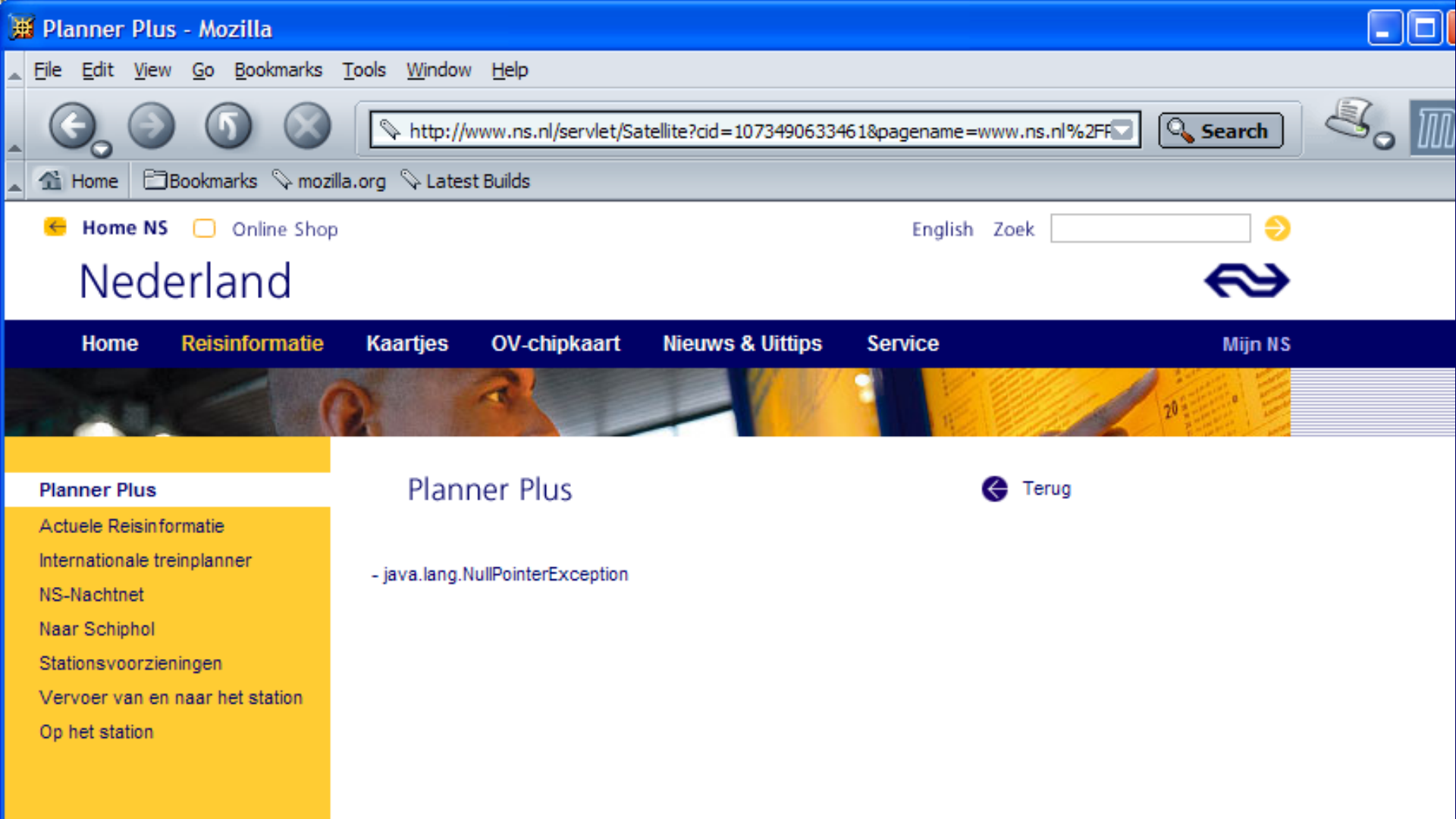
On Patterns and Pattern Languages

Volume 5

Frank Buschmann
Kevlin Henney
Douglas C. Schmidt


Jsing autodetected IRQ (11) to improve performance.
ifcust (PC/TCP Class 1 packet driver - DIX Ethernet) init
5 free packets of length 160, 5 free packets of length 1
The kernel is using asynchronous sends
The Resident Module occupies 0 bytes of conventional mem


PCI device listing ...
Bus No. Device No. Func No. Vendor/Device Class Device Class
0 7 1 1106 0571 0101 IDE Cntrlr
Jsing
ifcust
5 free
The ke
The Re
EUROPEAN UNION ONLY


Planner Plus - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://www.ns.nl/servlet/Satellite?cid=1073490633461&pagename=www.ns.nl%2FP
Search
Home Bookmarks mozilla.org Latest Builds
Home NS Online Shop
English Zoek
Nederland
Home Reisinformatie Kaartjes OV-chipkaart Nieuws & Uittips Service Mijn NS
Planner Plus
Actuele Reisinformatie
Internationale treinplanner
NS-Nachtnet
Naar Schiphol
Stationsvoorzieningen
Vervoer van en naar het station
Op het station
Planner Plus
Terug
- java.lang.NullPointerException

```
Error opening /com/hilton/hiway/portlets/reservation/book/BookReservationController.jpf.
The source of this error is:
com.bea.portlet.adapter.scopedcontent.ActionLookupFailedException: javax.servlet.ServletException: java.lang.ClassNotFoundException: handleUndefinedException
	at com.bea.portlet.adapter.scopedcontent.ScopedContentCommonSupport.executeAction(ScopedContentCommonSupport.java:697)
	at com.bea.portlet.adapter.scopedcontent.ScopedContentCommonSupport.renderInternal(ScopedContentCommonSupport.java:266)
	at com.bea.portlet.adapter.scopedcontent.PageFlowStubImpl.render(PageFlowStubImpl.java:136)
	at com.bea.netuix.servlets.controls.content.NetuiContent.preRender(NetuiContent.java:292)
	at com.bea.netuix.nf.ControlLifecycle$6.visit(ControlLifecycle.java:428)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:727)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walkRecursivePreRender(ControlTreeWalker.java:739)
	at com.bea.netuix.nf.ControlTreeWalker.walk(ControlTreeWalker.java:146)
	at com.bea.netuix.nf.Lifecycle.processLifecycles(Lifecycle.java:395)
	at com.bea.netuix.nf.Lifecycle.processLifecycles(Lifecycle.java:361)
	at com.bea.netuix.nf.Lifecycle.runOutbound(Lifecycle.java:208)
	at com.bea.netuix.nf.Lifecycle.run(Lifecycle.java:162)
	at com.bea.netuix.servlets.manager.UIServlet.runLifecycle(UIServlet.java:388)
	at com.bea.netuix.servlets.manager.UIServlet.processControlTree(UIServlet.java:301)
	at com.bea.netuix.servlets.manager.PortalServlet.service(PortalServlet.java:930)
	at javax.servlet.http.HttpServlet.service(HttpServlet.java:820)
	at weblogic.servlet.internal.StubSecurityHelper$ServletServiceAction.run(StubSecurityHelper.java:226)
	at weblogic.servlet.internal.StubSecurityHelper.invokeServlet(StubSecurityHelper.java:124)
	at weblogic.servlet.internal.ServletStubImpl.execute(ServletStubImpl.java:283)
	at weblogic.servlet.internal.TailFilter.doFilter(TailFilter.java:26)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.hilton.hiway.web.filters.CharsetFilter.doFilterInternal(CharsetFilter.java:18)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:75)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.hilton.hiway.web.filters.WebServicePayloadFilter.doFilterInternal(WebServicePayloadFilter.java:52)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:75)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.hilton.hiway.web.filters.NGTEchoFilter.doFilterInternal(NGTEchoFilter.java:46)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:75)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.hilton.hiway.web.filters.ParameterRemoverFilter.doFilter(ParameterRemoverFilter.java:46)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.hilton.hiway.web.filters.PerformanceMonitorFilter.doFilterInternal(PerformanceMonitorFilter.java:40)
	at org.springframework.web.filter.OncePerRequestFilter.doFilter(OncePerRequestFilter.java:75)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.bea.portal.tools.servlet.http.HttpContextFilter.doFilter(HttpContextFilter.java:60)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at com.bea.p13n.servlets.PortalServletFilter.doFilter(PortalServletFilter.java:336)
	at weblogic.servlet.internal.FilterChainImpl.doFilter(FilterChainImpl.java:42)
	at weblogic.servlet.internal.WebAppServletContext$ServletInvocationAction.run(WebAppServletContext.java:3402)
	at weblogic.security.acl.internal.AuthenticatedSubject.doAs(AuthenticatedSubject.java:321)
	at weblogic.security.service.SecurityManager.runAs(Unknown Source)
	at weblogic.servlet.internal.WebAppServletContext.securedExecute(WebAppServletContext.java:2140)
	at weblogic.servlet.internal.WebAppServletContext.execute(WebAppServletContext.java:2046)
	at weblogic.servlet.internal.ServletRequestImpl.run(ServletRequestImpl.java:1398)
	at weblogic.work.ExecuteThread.execute(ExecuteThread.java:200)
	at weblogic.work.ExecuteThread.run(ExecuteThread.java:172)
```

Need Help?

Email us with your enquiries.

Or Call
U.K. 0870 5 90 90 90
Outside U.K. 00800 4445 8667

Or visit our Customer Service page

Reservation Summary

Hotel

The Waldorf Hilton, London

Change Hotel

0000420002107603560O

EXPEDITED PARCEL
COLIS ACCÉLÉRÉS

2

CANADA POST / POSTES CANADA

From / Exp.:

$retAdd.getFirstName().toUpperCase()

$retAdd.getAddressLine1().toUpperCase()

$retAdd.getCity().toUpperCase() $retAdd.getState().toUpperCase() $retAdd.g

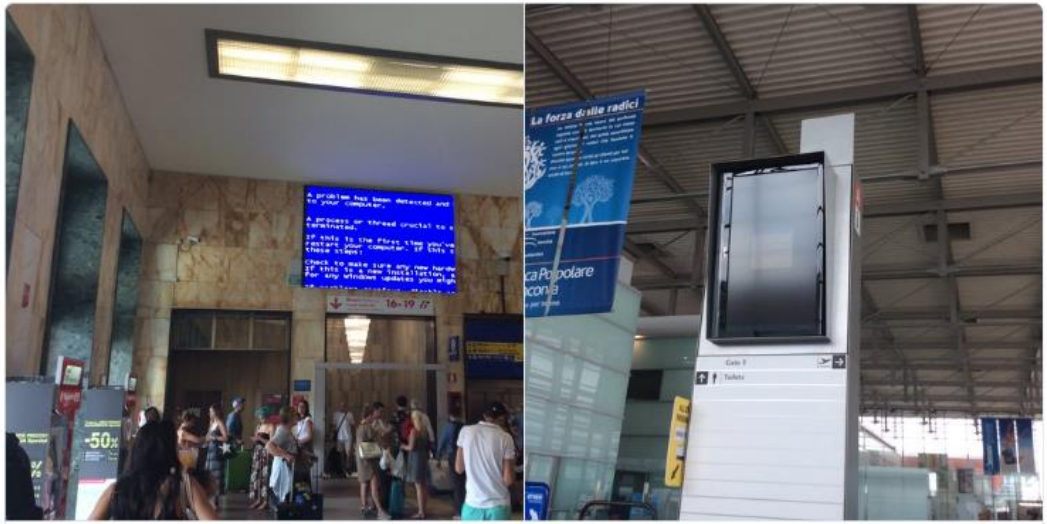
$retAdd.getDayPhone()

Payer / Facturé à:

7307904

Method of Payment /
Mode de paiement:

To / Dest.:



@tackline

Follow

Arriving in Bologna, I saw a @KevlinHenney screen. Whilst queueing to leave Ancona another appeared as I waited.

2:05 PM - 25 Jul 2016

3 2

https://twitter.com/tackline/status/757562488363843584

Gitte Klitgaard
@NativeWired

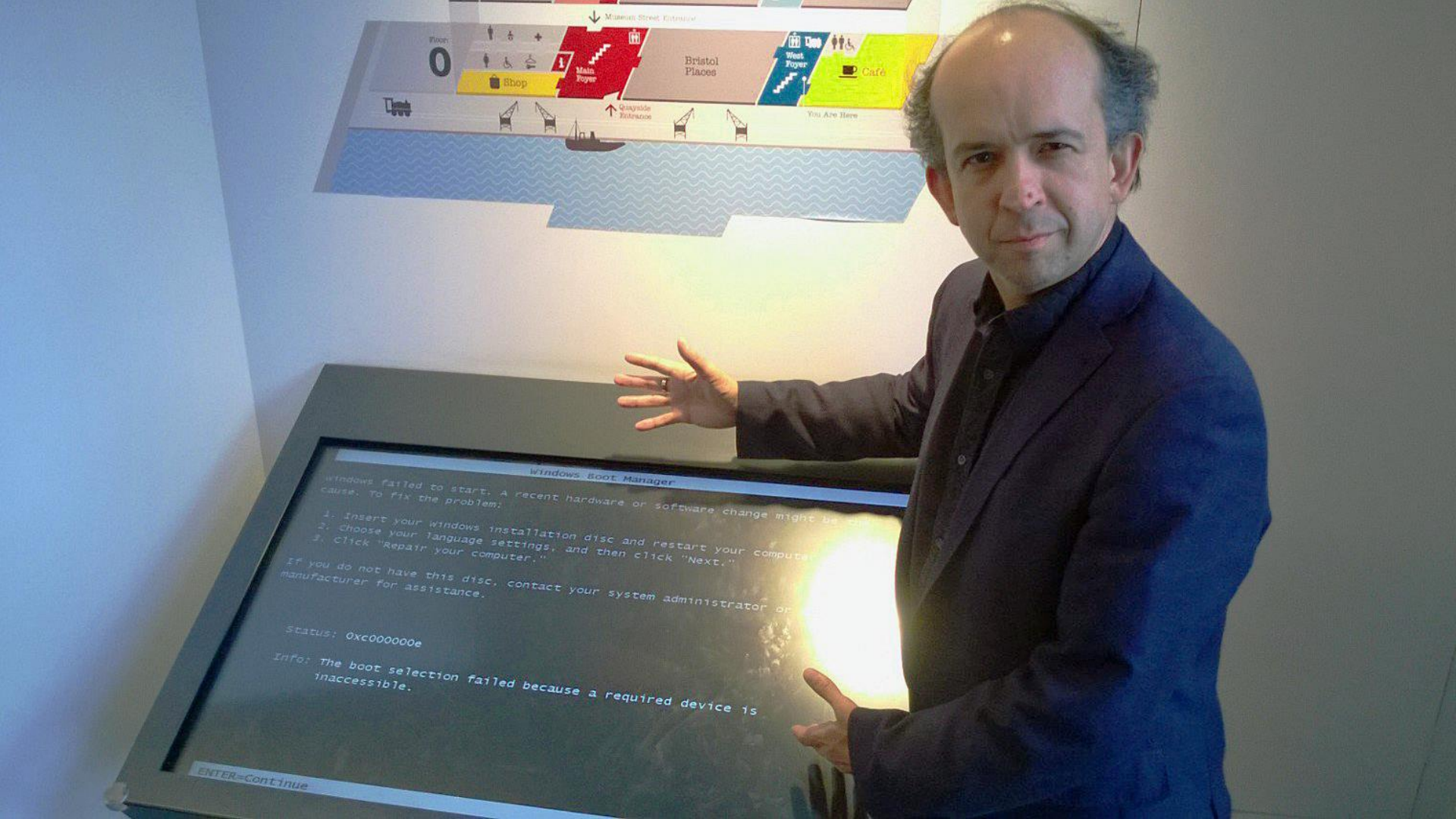
Follow

A @KevlinHenney at Copenhagen airport :)

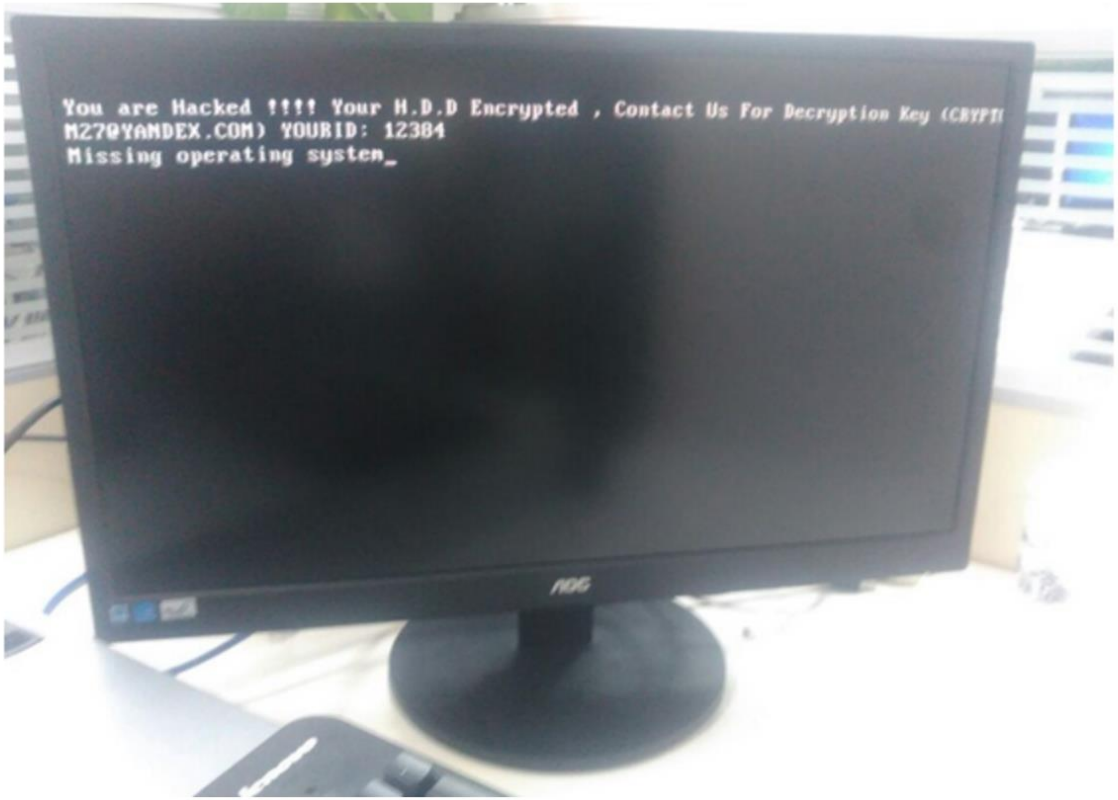
2:11 PM - 7 Feb 2017

4 5

https://twitter.com/NativeWired/status/828939258475999232

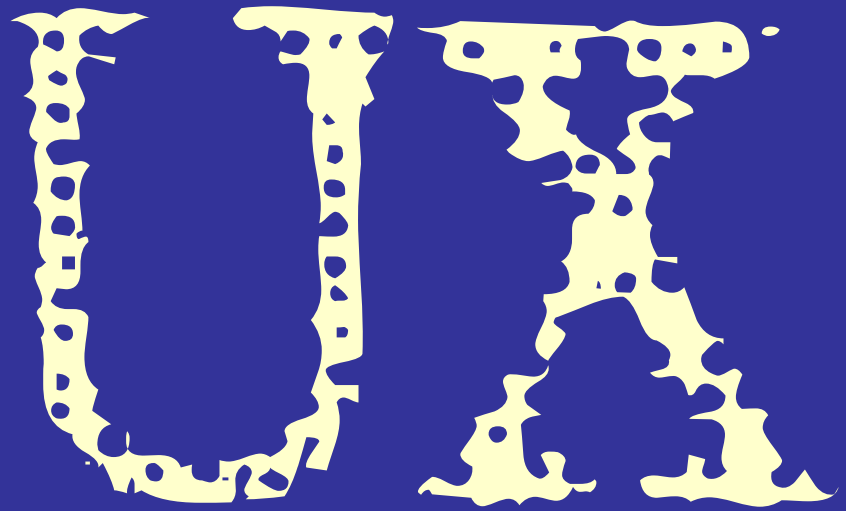

Floor
0
Shop
Main Foyer
Bristol Places
West Foyer
Café
Quayside Entrance
You Are Here
Windows Boot Manager
Windows failed to start. A recent hardware or software change might be
cause. To fix the problem:
1. Insert your windows installation disc and restart your computer
2. Choose your language settings, and then click "Next."
3. Click "Repair your computer."
If you do not have this disc, contact your system administrator or
manufacturer for assistance.
Status: 0xc000000e
Info: The boot selection failed because a required device is
inaccessible.
ENTER=Continue

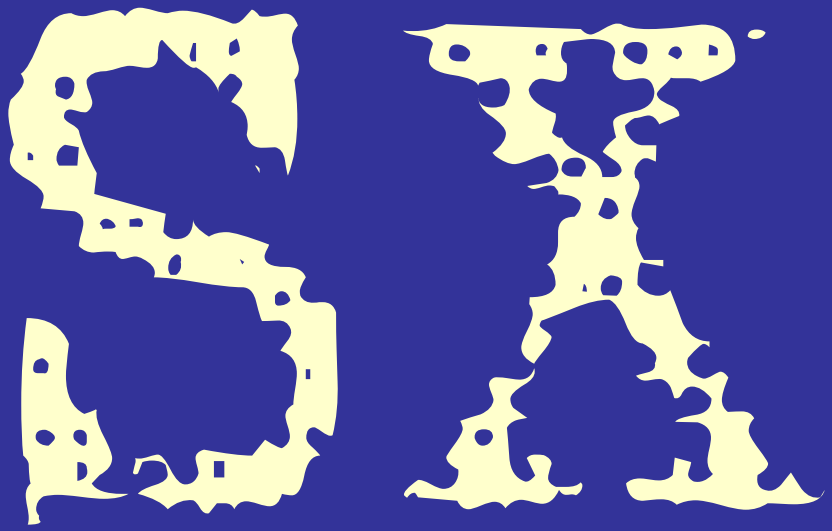

Abfahrt
Linie
Ziel
Gleis
Zeit
Über
Nach
Gleis
22:10 RB81
Flöha - Pockau-Lengefeld
Olbernhau
8
22:30 RB30
Flöha - Frei
- Fährt heut
Hbf
11
22:31 RB30
Hohenstein
(S) Hbf
10
22:36 RB80
Flöha - Zsc
g-B. Süd
8
22:36 RB45
irt heute von
Geithain - E
9
22:44 RE6
Einsiedel - Thalheim (Erzgeb)
lbf
5
22:45 RB89
Flöha - Freiberg (Sachs) - Tharandt
Aue (Sachs)
14
23:30
Fährt heute von Gleis 11 -
Dresden Hbf
11
Ooops, your files have been encrypted!
Was geschah mit meinem Computer?
Kann ich meine Dateien wiederherstellen?
Wie bezahle ich?
Payment will be raised on
5/15/2017 22:50:58
Time Left
02:23:18:55
Your files will be lost on
5/19/2017 22:50:58
Time Left
06:23:18:55
About bitcoin
How to buy bitcoins?
Contact Us
bitcoin
ACCEPTED HERE
Send $300 worth of bitcoin to this address:
12dYUPgeusZ9NyMgw519p7AA8isjr6SMw
Copy
Check Payment
Decrypt
BMG | MIS

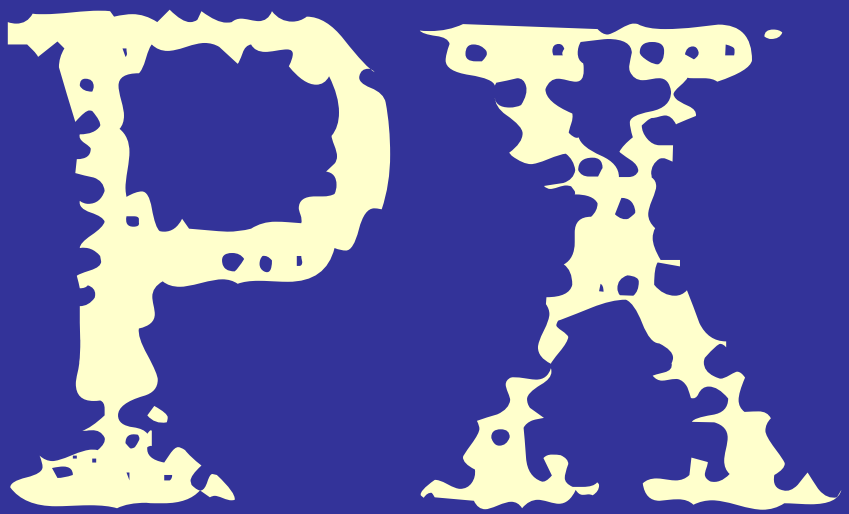


A copy of the ransom message left behind by the "Mamba" ransomware.

On Friday, *The San Francisco Examiner* reported that riders of SFMTA's Municipal Rail or "Muni" system were greeted with handmade "Out of Service" and "Metro Free" signs on station ticket machines. The computer terminals at all Muni locations carried the "hacked" message: "Contact for key (**cryptom27@yandex.com**)," the message read.


UX


Sx


PX

```
if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;
```

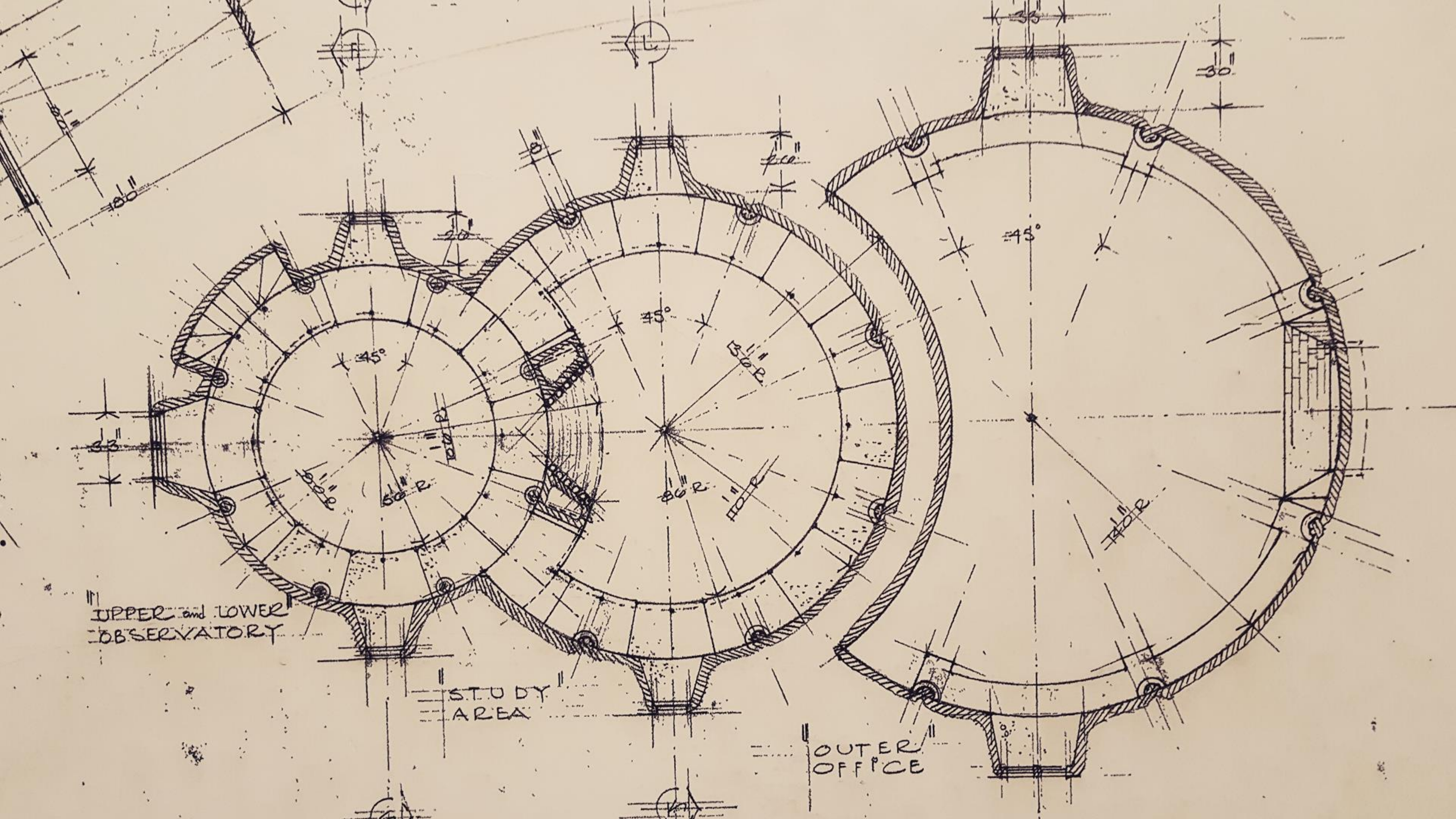
Mike Bland
"Goto Fail, Heartbleed, and Unit Testing Culture"
https://martinfowler.com/articles/testing-culture.html

```
network code()
{
    switch (line) {
    case THING1:
        doit1();
        break;
    case THING2:
        if (x == STUFF) {
           do_first_stuff();
           if (y == OTHER_STUFF)
                break;
           do_later_stuff();
        } /* coder meant to break to here... */
        initialize_modes_pointer();
        break;
    default:
        processing();
    } /* ...but actually broke to here! */
    use_modes_pointer(); /* leaving the modes_pointer uninitialized */
}
```

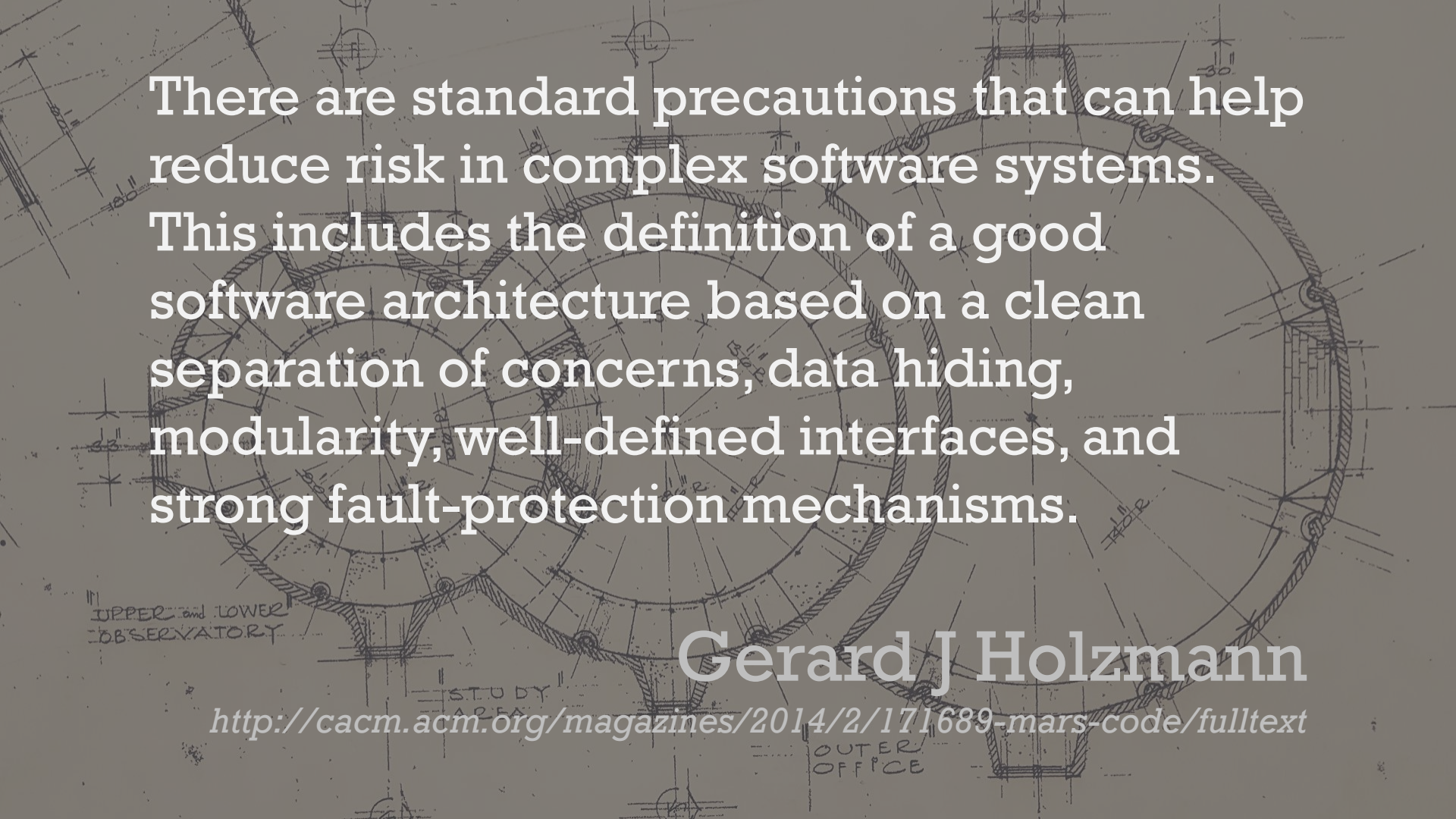
Peter van der Linden
Expert C Programming

Most of our systems are much more complicated than can be considered healthy, and are too messy and chaotic to be used in comfort and confidence.

Edsger W Dijkstra



UPPER and LOWER
OBSERVATORY
STUDY
AREA
OUTER
OFFICE
45°
45°
45°



There are standard precautions that can help reduce risk in complex software systems. This includes the definition of a good software architecture based on a clean separation of concerns, data hiding, modularity, well-defined interfaces, and strong fault-protection mechanisms.

Gerard J Holzmann

http://cacm.acm.org/magazines/2014/2/171689-mars-code/fulltext


Adam Jacot de Boinod
I NEVER KNEW THERE WAS A WORD FOR IT
'Very funny'
STEPHEN FRY
WORDSWORTH REFERENCE
The Wordsworth Book of Intriguing Words
The insomniac's dictionary of the outrageous, odd and unusual
Paul Hellweg
REFERENCE
CONCISE OXFORD DICTIONARY OF
English
mology
T. F. HOAD
word origins
the language report
Joie de vivre n.
COMPILED BY
A COMPREHENSIVE WORD-FINDING DICTIONARY
Chambers
BILL
BRYSON
TROUBLESOME WORDS
'Combines the virtues of a first-class work of reference with the pleasure of a good read' The Times
FULLY REVISED AND UPDATED
Oxford
Samuel Johnson's
LOCK
COLLINS
REFERENCE DICTIONARY
MATHEMATICS
E.J. BOROWSKI AND J.M. BORWEIN
WORDS
ME
EFFREY KACIRK
AUTHOR OF Forgotten English

/ WordFriday

code, *noun*

- a set of instructions for a computer
- a computer program, or a portion thereof
- a system of words, figures or symbols used to represent others, especially for the purposes of secrecy
- a set of conventions or principles governing behaviour or activity in a particular domain

Concise Oxford English Dictionary · Oxford English Dictionary · Merriam-Webster's Collegiate Dictionary

risk, *noun*

- a situation involving exposure to danger
- the chance or hazard of commercial loss
- product of the consequence and probability of a hazardous event or phenomenon
- exposure to a proposition of which one is uncertain

Concise Oxford English Dictionary · Oxford English Dictionary · Wikipedia · "Defining Risk" by Glen A Holton

kcpeppe
@kcpeppe

@KevlinHenney functionality is an asset, code is a liability

8:14 AM - 5 Jun 2010

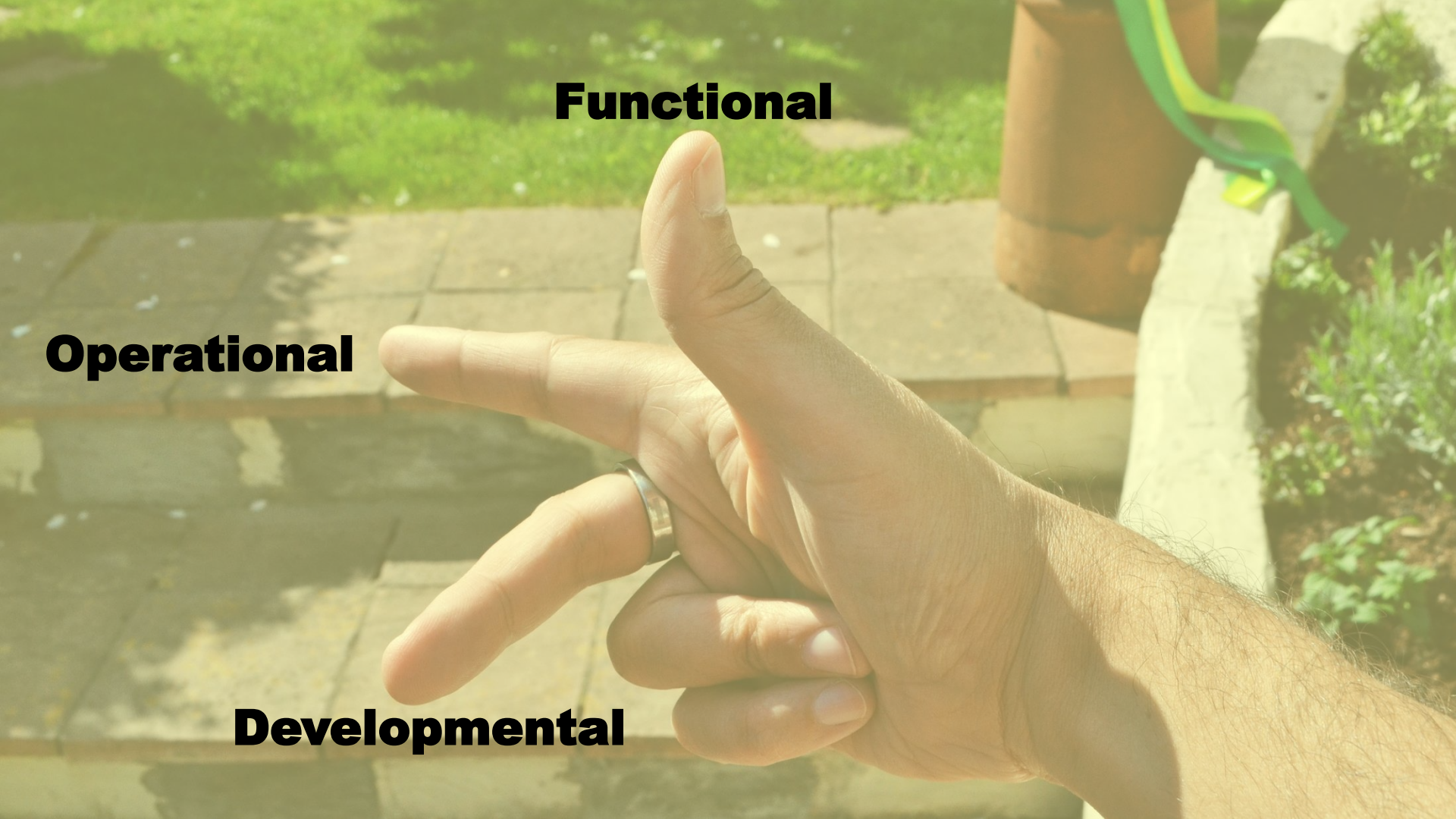
4 4

https://twitter.com/kcpeppe/status/15473004648

Avoiding complexity reduces bugs.

Linus Torvalds

Avoiding complexity reduces vulnerabilities.


Functional
Operational
Developmental

```
Connection * CreateServerConnection()
{
    // Declarations
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;

    // Get address and check that its OK (throw an exception if its not)
    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Convert adress to bytes and check that its OK (throw an exception if its not)
    address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Get port and check that its OK (throw an exception if its not)
    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Convert port too bytes
    port = htons(atoi(cfgPort.data()));

    // Creation connection and check that its OK (throw an exception if its not)
    result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Return the connection
    return result;
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;

    // Get address and check that its OK (throw an exception if its not)
    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Convert adress to bytes and check that its OK (throw an exception if its not)
    address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Get port and check that its OK (throw an exception if its not)
    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Convert port too bytes
    port = htons(atoi(cfgPort.data()));

    // Creation connection and check that its OK (throw an exception if its not)
    result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    // Return the connection
    return result;
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Get address and check that its OK (throw an exception if its not)
    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Convert adress to bytes and check that its OK (throw an exception if its not)
    address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Get port and check that its OK (throw an exception if its not)
    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Convert port too bytes
    port = htons(atoi(cfgPort.data()));
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Creation connection and check that its OK (throw an exception if its not)
    result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    // Return the connection
    return result;
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    ...
    // Get address and check that its OK (throw an exception if its not)
    ...
    // Convert adress to bytes and check that its OK (throw an exception if its not)
    ...
    // Get port and check that its OK (throw an exception if its not)
    ...
    // Convert port too bytes
    ...
    // Creation connection and check that its OK (throw an exception if its not)
    ...
    // Return the connection
    ...
}
```

```
Connection * CreateServerConnection()
{
    // Declarations
    ...
    // Get address and check that it's OK (throw an exception if it's not)
    ...
    // Convert address to bytes and check that it's OK (throw an exception if it's not)
    ...
    // Get port and check that it's OK (throw an exception if it's not)
    ...
    // Convert port to bytes
    ...
    // Creation connection and check that it's OK (throw an exception if it's not)
    ...
    // Return the connection
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    ...
    ...
    ...
    ...
    ...
    ...
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];
    std::string cfgAddress;
    unsigned long address;
    std::string cfgPort;
    unsigned short port;
    Connection * result;

    cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    port = htons(atoi(cfgPort.data()));

    result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    std::string cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    unsigned long address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    std::string cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    unsigned short port = htons(atoi(cfgPort.data()));

    Connection * result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.data()));

    Connection * result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    ...
    Connection * result = new Connection(address, port);
    if (!result || !result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    ...
    Connection * result = new Connection(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
std::auto_ptr<Connection> CreateServerConnection()
{
    ...
    std::auto_ptr<Connection> result(new Connection(address, port));
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
std::unique_ptr<Connection> CreateServerConnection()
{
    ...
    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result;
}
```

```
Connection * CreateServerConnection()
{
    ...
    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.data()));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.data());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.data()));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.data(), cfgPort.data());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

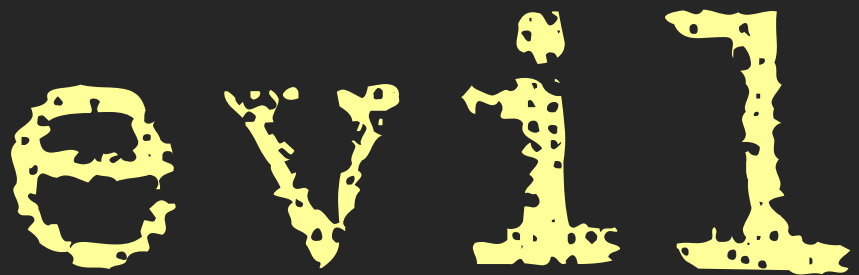
```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(atoi(cfgPort.c_str()));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

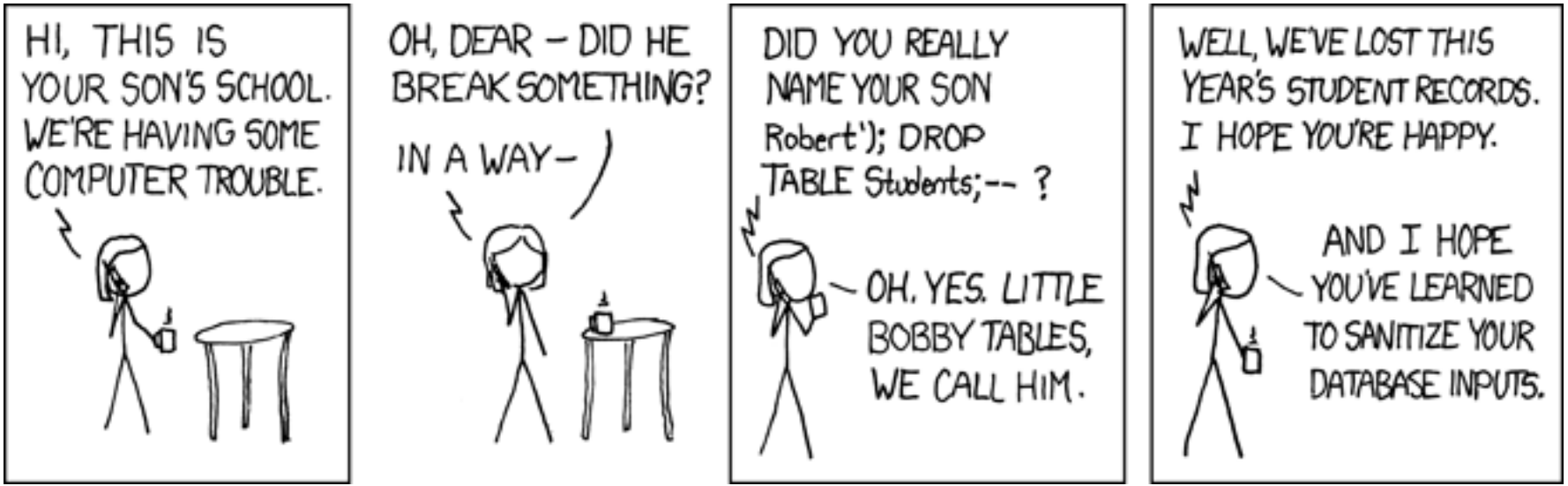
```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

printf


eval


evil



https://xkcd.com/327/

Every escape is an entrance

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        sprintf(buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        sprintf(buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        sprintf(buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];

    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        snprintf(buffer, sizeof buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        snprintf(buffer, sizeof buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        snprintf(buffer, sizeof buffer, "Configuration value missing: %s", "port");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        snprintf(buffer, sizeof buffer, "Failed to connect: %s:%s", cfgAddress.c_str(), cfgPort.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    char buffer[1024];
    ...
    if (cfgAddress.empty())
    {
        snprintf(buffer, sizeof buffer, "Configuration value missing: %s", "address");
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
    if (address == -1)
    {
        snprintf(buffer, sizeof buffer, "Invalid address: %s", cfgAddress.c_str());
        Log::Instance().Write(buffer);
        throw ConnectionException(buffer);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    if (cfgAddress.empty())
    {
        std::stringstream buffer;
        buffer << "Configuration value missing: " << "address";
        Log::Instance().Write(buffer.str());
        throw ConnectionException(buffer.str());
    }
    ...
    if (address == -1)
    {
        std::stringstream buffer;
        buffer << "Invalid address: " << cfgAddress;
        Log::Instance().Write(buffer.str());
        throw ConnectionException(buffer.str());
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    ...
    if (cfgAddress.empty())
    {
        static const char * logMessage = "Configuration value missing: address";
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }
    ...
    if (address == -1)
    {
        auto logMessage = "Invalid address: " + cfgAddress;
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }
    ...
}
```

```
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        static const char * logMessage = "Configuration value missing: address";
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        auto logMessage = "Invalid address: " + cfgAddress;
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        static const char * logMessage = "Configuration value missing: port");
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        auto logMessage = "Failed to connect: " + cfgAddress + ":" + cfgPort;
        Log::Instance().Write(logMessage);
        throw ConnectionException(logMessage);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
    {
        FailedToConnect("Configuration value missing: address");
    }

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
    {
        FailedToConnect("Invalid address: " + cfgAddress);
    }

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
    {
        FailedToConnect("Configuration value missing: port");
    }

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
    {
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);
    }

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result.release();
}
```

```
Connection * CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result.release();
}
```

```
std::unique_ptr<Connection> CreateServerConnection()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer()
{
    auto cfgAddress = ConfigurationManager::Instance().GetValue("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().GetValue("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer()
{
    auto cfgAddress = ConfigurationManager::Instance().ValueOf("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = ConfigurationManager::Instance().ValueOf("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer()
{
    auto cfgAddress = Configuration::Instance().ValueOf("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = Configuration::Instance().ValueOf("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```


Share a Coke with
Kevlin
FROME VALLEY
HEREFORDSHIRE
HENNEY'S
DRY CIDER
MADE FROM 100% FRESH PRESSED JUICE
AGED 12 YEARS
OF DUFFTOWN
TRADE MARK
SINGLETON
of Dufftown
STD 1896
12
PRODUCT OF SCOTLAND

Early Detection of Configuration Errors to Reduce Failure Damage

https://www.usenix.org/system/files/conference/osdi16/osdi16-xu.pdf

Our study shows that many of today's mature, widely used software systems are subject to latent configuration errors in their critically important configurations.

https://www.usenix.org/system/files/conference/osdi16/osdi16-xu.pdf

One root cause is that many (14.0%–93.2%) of these configurations do not have any special code for checking the correctness of their settings at the system's initialization time.

https://www.usenix.org/system/files/conference/osdi16/osdi16-xu.pdf

```
std::unique_ptr<Connection> ConnectToServer()
{
    auto cfgAddress = Configuration::Instance().ValueOf("address");
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    auto cfgPort = Configuration::Instance().ValueOf("port");
    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer(
    const std::string & cfgAddress, const std::string & cfgPort)
{
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

Be conservative in what you do, be liberal in what you accept from others.

Postel's law

Be conservative in what you do, be conservative in what you accept from others.

```
std::unique_ptr<Connection> ConnectToServer(
    const std::string & cfgAddress, const std::string & cfgPort)
{
    if (cfgAddress.empty())
        FailedToConnect("Configuration value missing: address");

    auto address = inet_addr(cfgAddress.c_str());
    if (address == -1)
        FailedToConnect("Invalid address: " + cfgAddress);

    if (cfgPort.empty())
        FailedToConnect("Configuration value missing: port");

    auto port = htons(stoi(cfgPort));

    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect("Failed to connect: " + cfgAddress + ":" + cfgPort);

    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer(in_addr_t address, in_port_t port)
{
    auto result = std::make_unique<Connection>(address, port);
    if (!result->IsOK())
        FailedToConnect(address, port);
    return result;
}
```

```
std::unique_ptr<Connection> ConnectToServer(in_addr_t address, in_port_t port)
{
    return std::make_unique<Connection>(address, port);
}
```

Remember that there is no code faster than no code.

Taligent's Guide to Designing Programs

Remember that there is no code more secure than no code.

How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript

Code pulled from NPM – which everyone was using

Careful, careful ... Don't fumble this like the JS world (Credit: Claus Rebler)

23 Mar 2016 at 01:24, Chris Williams 1322

Updated Programmers were left staring at broken builds and failed installations on Tuesday after someone toppled the Jenga tower of JavaScript.

A couple of hours ago, Azer Koçulu unpublished more than 250 of his modules from NPM, which is a popular package manager used by JavaScript projects to install dependencies.

http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

```
function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

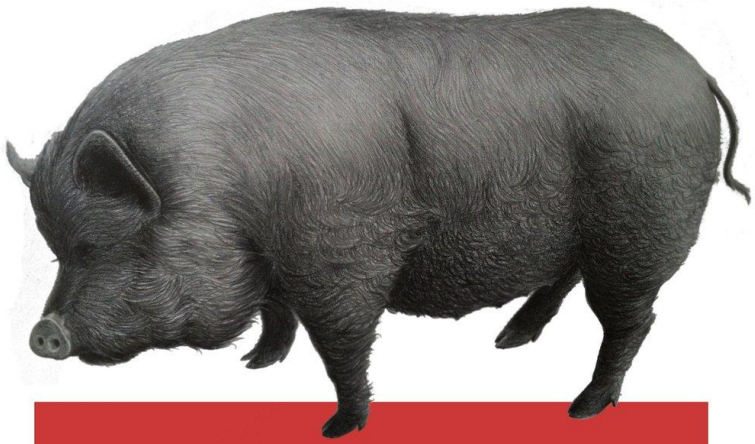
```
var cache = [
  '',
  ' ',
  '  ',
  '   ',
  '    ',
  '     ',
  '      ',
  '       ',
  '        ',
  '         '
];

function leftPad (str, len, ch) {
  // convert `str` to `string`
  str = str + '';
  // `len` is the `pad`'s length now
  len = len - str.length;
  // doesn't need to pad
  if (len <= 0) return str;
  // `ch` defaults to `' '`
  if (!ch && ch !== 0) ch = ' ';
  // convert `ch` to `string`
  ch = ch + '';
  // cache common use cases
  if (ch === ' ' && len < 10) return cache[len] + str;
  // `pad` starts with an empty string
  var pad = '';
  // loop
  while (true) {
    // add `ch` to `pad` if `len` is odd
    if (len & 1) pad += ch;
    // divide `len` by 2, ditch the remainder
    len >>= 1;
    // "double" the `ch` so this operation count grows logarithmically on `len`
    // each time `ch` is "doubled", the `len` would need to be "doubled" too
    // similar to finding a value in binary search tree, hence O(log(n))
    if (len) ch += ch;
    // `len` is 0, exit the loop
    else break;
  }
  // pad `str`!
  return pad + str;
}
```

https://github.com/stevemao/left-pad/blob/master/index.js

I have yet to see any problem, however complicated, which, when you looked at it in the right way, did not become still more complicated.

Anderson's Law

Code written by some stranger on the internet is always perfect



Taking on Needless Dependencies

Fragile Development Guide

O RLY?

@ThePracticalDev

Laurie Voss
@seldo

Hey npm users: left-pad 0.0.3 was unpublished, breaking LOTS of builds. To fix, we are un-un-publishing it at the request of the new owner.

12:03 AM - 23 Mar 2016

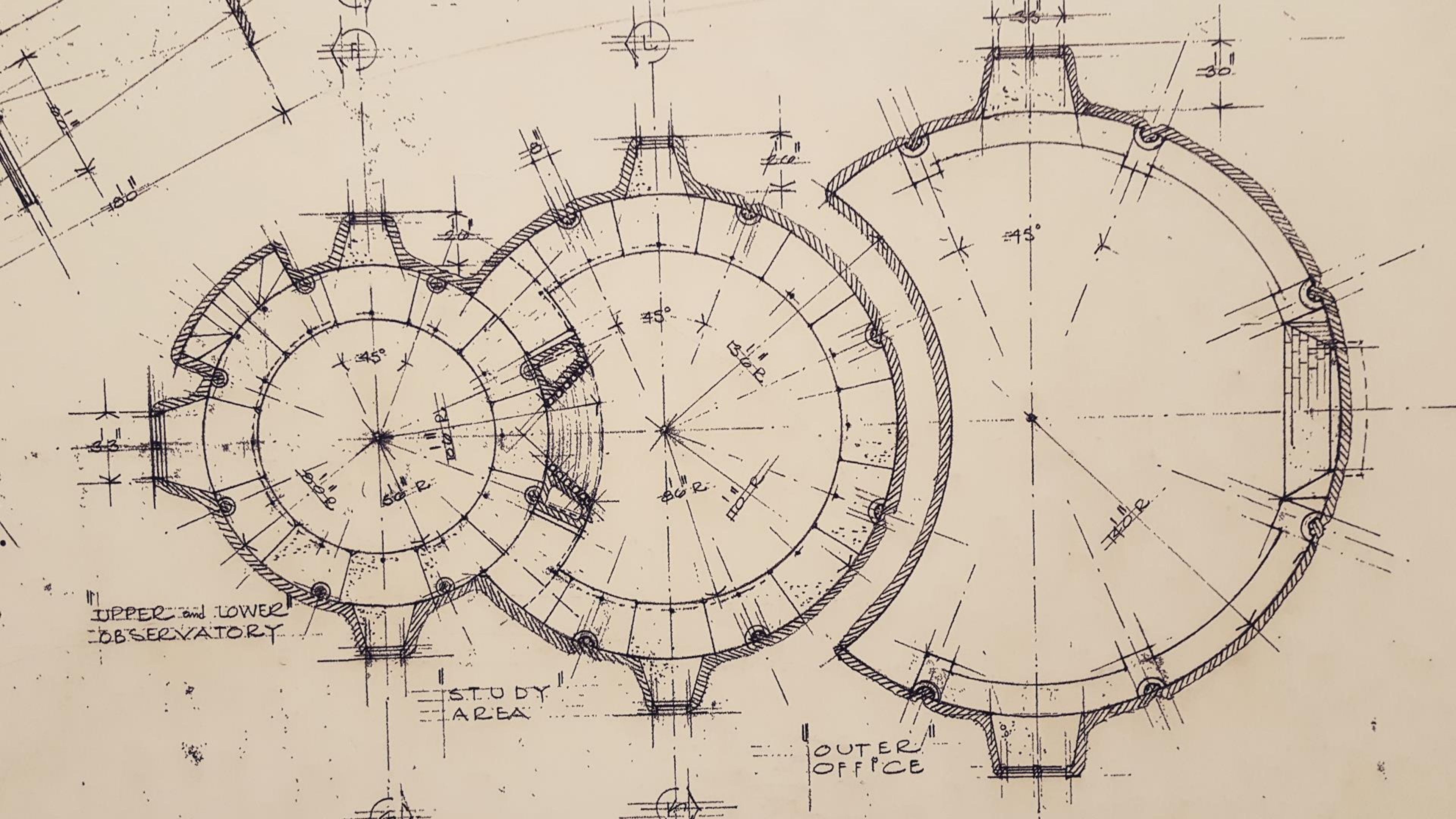
274 228

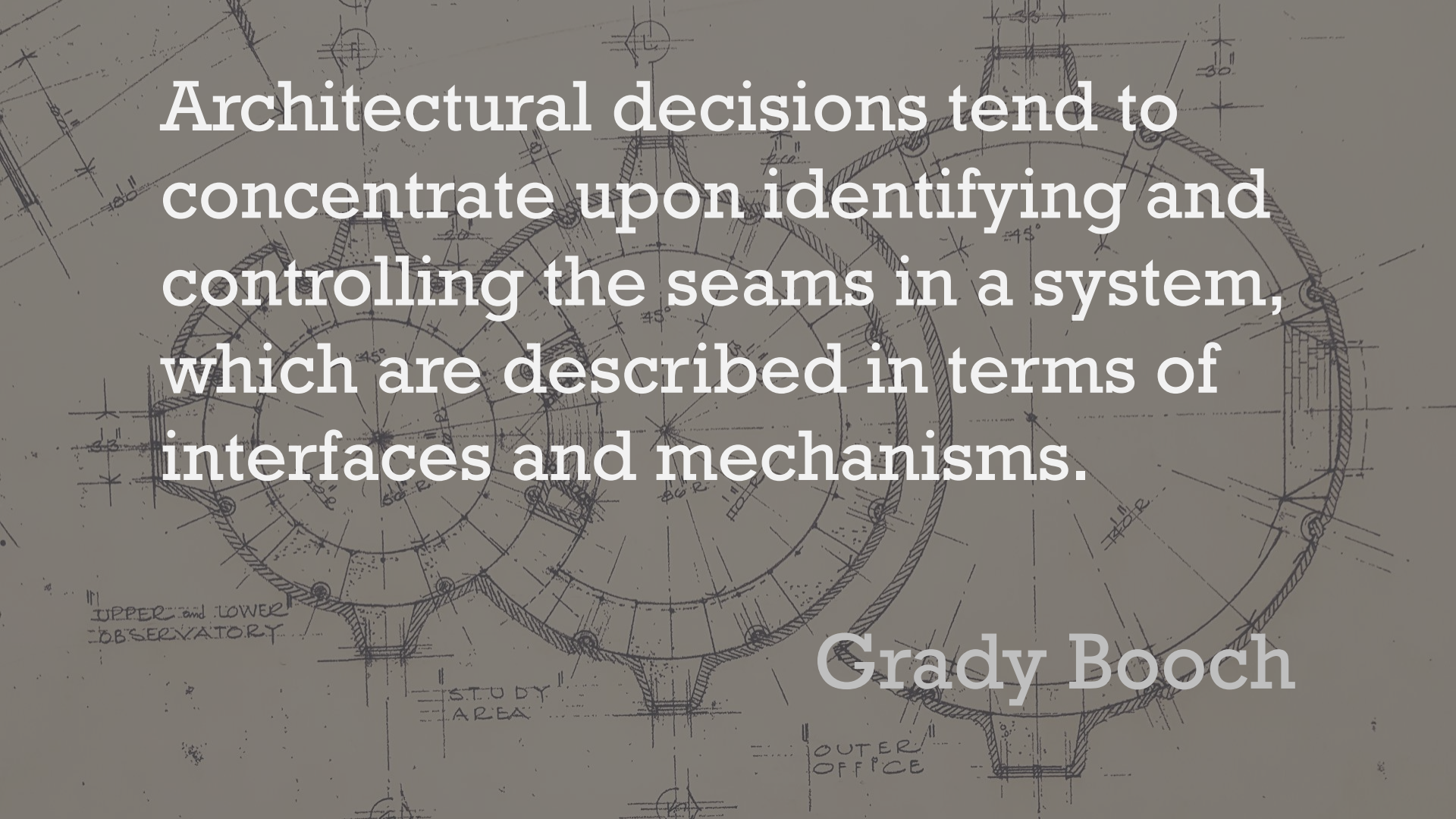
https://twitter.com/seldo/status/712414400808755200

```
function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

```
function leftpad (str, len, ch) {
  somethingWickedThisWayComes()
  return _leftpad(str, len, ch);
}
```


UPPER and LOWER OBSERVATORY
STUDY AREA
OUTER OFFICE
45°
45°
45°
33"
33"
30"



Architectural decisions tend to concentrate upon identifying and controlling the seams in a system, which are described in terms of interfaces and mechanisms.

Grady Booch

As mankind relies more and more on the software that controls the computers that in turn guide society, it becomes crucial that people control absolutely the programs and the processes by which they are produced, throughout the useful life of the program.

Meir M Lehman

"Programs, Life Cycles, and Laws of Software Evolution"

Goto Fail, Heartbleed, and Unit Testing Culture

Mike Bland

https://martinfowler.com/articles/testing-culture.html

These bugs are as instructive as they were devastating: They were rooted in the same programmer optimism, overconfidence, and haste that strike projects of all sizes and domains.

Mike Bland
https://martinfowler.com/articles/testing-culture.html

These bugs arouse my passion because I've seen and lived the benefits of unit testing, and this strongly-imprinted experience compels me to reflect on how unit testing approaches could prevent defects as high-impact and high-profile as these SSL bugs.

Mike Bland
https://martinfowler.com/articles/testing-culture.html

```
function leftpad (str, len, ch) {
  str = String(str);

  var i = -1;

  if (!ch && ch !== 0) ch = ' ';

  len = len - str.length;

  while (++i < len) {
    str = ch + str;
  }

  return str;
}
```

http://www.theregister.co.uk/2016/03/23/npm_left_pad_chaos/

```
var cache = [
  '',
  ' ',
  '  ',
  '   ',
  '    ',
  '     ',
  '      ',
  '       ',
  '        ',
  '         '
];

function leftPad (str, len, ch) {
  // convert `str` to `string`
  str = str + '';
  // `len` is the `pad`'s length now
  len = len - str.length;
  // doesn't need to pad
  if (len <= 0) return str;
  // `ch` defaults to `' '`
  if (!ch && ch !== 0) ch = ' ';
  // convert `ch` to `string`
  ch = ch + '';
  // cache common use cases
  if (ch === ' ' && len < 10) return cache[len] + str;
  // `pad` starts with an empty string
  var pad = '';
  // loop
  while (true) {
    // add `ch` to `pad` if `len` is odd
    if (len & 1) pad += ch;
    // divide `len` by 2, ditch the remainder
    len >>= 1;
    // "double" the `ch` so this operation count grows logarithmically on `len`
    // each time `ch` is "doubled", the `len` would need to be "doubled" too
    // similar to finding a value in binary search tree, hence O(log(n))
    if (len) ch += ch;
    // `len` is 0, exit the loop
    else break;
  }
  // pad `str`!
  return pad + str;
}
```

https://github.com/stevemao/left-pad/blob/master/index.js

```
function leftpad(content, length, pad) {
  content = String(content)
  pad = String(pad || pad === 0 ? pad : ' ')[0]
  var left = Math.max(length - content.length, 0)
  return pad.repeat(left) + content
}
```

```
truths = {
    "Padding an empty string to a length of 0 results in an empty string":
        leftpad("", 0, "X") === "",
    "Padding a non-empty string to a shorter length results in the same string":
        leftpad("foobar", 3, "X") === "foobar",
    "Padding a non-empty string to a negative length results in the same string":
        leftpad("foobar", -3, "X") === "foobar",
    "Padding a non-empty string to its length results in the same string":
        leftpad("foobar", 6, "X") === "foobar",
    "Padding to a longer length with a single character fills to the left":
        leftpad("foobar", 8, "X") === "XXfoobar",
    "Padding to a longer length with surplus characters fills using only first":
        leftpad("foobar", 10, "XY") === "XXXXfoobar",
    "Padding to a longer length with an empty string fills with space":
        leftpad("foobar", 8, "") === "  foobar",
    "Padding to a longer length with no specified fill fills with space":
        leftpad("foobar", 9) === "   foobar",
    "Padding to a longer length with integer 0 fills with 0":
        leftpad("foobar", 7, 0) === "0foobar",
    "Padding to a longer length with single-digit integer fills with digit":
        leftpad("foobar", 10, 1) === "1111foobar",
    "Padding to a longer length with multiple-digit integer fills with first digit":
        leftpad("foobar", 10, 42) === "4444foobar",
    "Padding to a longer length with negative integer fills with -":
        leftpad("foobar", 8, -42) === "--foobar",
    "Padding a non-string uses string representation":
        leftpad(4.2, 5, 0) === "004.2",
}
```

```
truths = {
    "Padding an empty string to a length of 0 results in an empty string":
        leftpad("", 0, "X") === "",
    "Padding a non-empty string to a shorter length results in the same string":
        leftpad("foobar", 3, "X") === "foobar",
    "Padding a non-empty string to a negative length results in the same string":
        leftpad("foobar", -3, "X") === "foobar",
    "Padding a non-empty string to its length results in the same string":
        leftpad("foobar", 6, "X") === "foobar",
    "Padding to a longer length with a single character fills to the left":
        leftpad("foobar", 8, "X") === "XXfoobar",
    "Padding to a longer length with surplus characters fills using only first":
        leftpad("foobar", 10, "XY") === "XXXXfoobar",
    "Padding to a longer length with an empty string fills with space":
        leftpad("foobar", 8, "") === "  foobar",
    "Padding to a longer length with no specified fill fills with space":
        leftpad("foobar", 9) === "   foobar",
    "Padding to a longer length with integer 0 fills with 0":
        leftpad("foobar", 7, 0) === "0foobar",
    "Padding to a longer length with single-digit integer fills with digit":
        leftpad("foobar", 10, 1) === "1111foobar",
    "Padding to a longer length with multiple-digit integer fills with first digit":
        leftpad("foobar", 10, 42) === "4444foobar",
    "Padding to a longer length with negative integer fills with -":
        leftpad("foobar", 8, -42) === "--foobar",
    "Padding a non-string uses string representation":
        leftpad(4.2, 5, 0) === "004.2",
}
```

```
truths = {
    "Padding an empty string to a length of 0 results in an empty string":
        leftpad("", 0, "X") === "",
    "Padding a non-empty string to a shorter length results in the same string":
        leftpad("foobar", 3, "X") === "foobar",
    "Padding a non-empty string to a negative length results in the same string":
        leftpad("foobar", -3, "X") === "foobar",
    "Padding a non-empty string to its length results in the same string":
        leftpad("foobar", 6, "X") === "foobar",
    "Padding to a longer length with a single character fills to the left":
        leftpad("foobar", 8, "X") === "XXfoobar",
    "Padding to a longer length with surplus characters fills using only first":
        leftpad("foobar", 10, "XY") === "XXXXfoobar",
    "Padding to a longer length with an empty string fills with space":
        leftpad("foobar", 8, "") === "  foobar",
    "Padding to a longer length with no specified fill fills with space":
        leftpad("foobar", 9) === "   foobar",
    "Padding to a longer length with integer 0 fills with 0":
        leftpad("foobar", 7, 0) === "0foobar",
    "Padding to a longer length with single-digit integer fills with digit":
        leftpad("foobar", 10, 1) === "1111foobar",
    "Padding to a longer length with multiple-digit integer fills with first digit":
        leftpad("foobar", 10, 42) === "4444foobar",
    "Padding to a longer length with negative integer fills with -":
        leftpad("foobar", 8, -42) === "--foobar",
    "Padding a non-string uses string representation":
        leftpad(4.2, 5, 0) === "004.2",
}
```

```
toMap = object => new Map(Object.entries(object))

format = (proposition, ok) =>
  proposition.fontcolor(ok ? "green" : "red") + "<br>"

present = truths =>
  toMap(truths).forEach(
    (ok, proposition) => write(format(proposition, ok)))

present(truths)
```

Padding an empty string to a length of 0 results in an empty string
Padding a non-empty string to a shorter length results in the same string
Padding a non-empty string to a negative length results in the same string
Padding a non-empty string to its length results in the same string
Padding to a longer length with a single character fills to the left
Padding to a longer length with surplus characters fills using only first
Padding to a longer length with an empty string fills with space
Padding to a longer length with no specified fill fills with space
Padding to a longer length with integer 0 fills with 0
Padding to a longer length with single-digit integer fills with digit
Padding to a longer length with multiple-digit integer fills with first digit
Padding to a longer length with negative integer fills with -
Padding a non-string uses string representation

Padding an empty string to a length of 0 results in an empty string

Padding a non-empty string to a shorter length results in the same string

Padding a non-empty string to a negative length results in the same string

Padding a non-empty string to its length results in the same string

Padding to a longer length with a single character fills to the left

Padding to a longer length with surplus characters fills using only first

Padding to a longer length with an empty string fills with space

Padding to a longer length with no specified fill fills with space

Padding to a longer length with integer 0 fills with 0

Padding to a longer length with single-digit integer fills with digit

Padding to a longer length with multiple-digit integer fills with first digit

Padding to a longer length with negative integer fills with -

Padding a non-string uses string representation

Testing Is the Engineering Rigor of Software Development

Neal Ford

passive

POUT

Plain
Ol'
Unit
Testing

POUT

active

POUT

TDD

Test-
Driven
Development

POUT

TDD

reactive

POUT

TDD

DDT

Defect-
Driven
Testing

Simple Testing Can Prevent Most Critical Failures

An Analysis of Production Failures in Distributed Data-Intensive Systems

https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-yuan.pdf

Almost all catastrophic failures are the result of incorrect handling of non-fatal errors explicitly signalled in software.

https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-yuan.pdf

A majority of the production failures (77%) can be reproduced by a unit test.

https://www.usenix.org/system/files/conference/osdi14/osdi14-paper-yuan.pdf

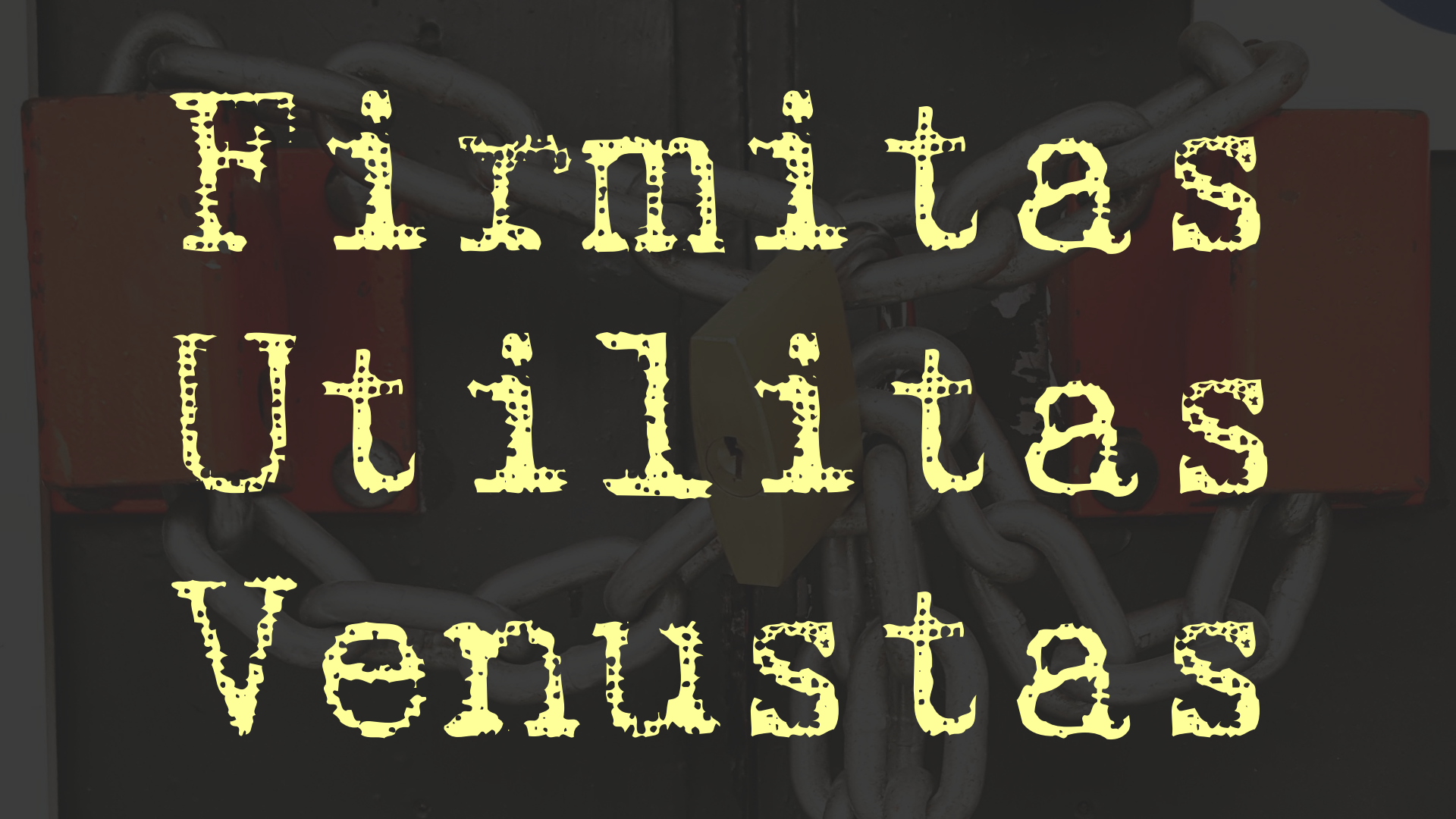
Top 10 Secure Coding Practices

1. Validate input
2. Heed compiler warnings
3. Architect and design for security policies
4. Keep it simple
5. Default deny
6. Adhere to the principle of least privilege
7. Sanitize data sent to other systems
8. Practice defense in depth
9. Use effective quality assurance techniques
10. Adopt a secure coding standard

Bonus Secure Coding Practices

1. Define security requirements
2. Model threats

https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices


Firmitas
Utilitas
Venustas