accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Is Git evil?

Charles Bailey

11th April 2013

git add

- But I've already added my file!

git add

- But I've already added my file!

EVIL!

git bisect

- `git bisect start <bad> <good>`

git bisect

- `git bisect start <bad> <good>`
- Have you ever had to try and find where something was fixed?

git bisect

- `git bisect start <bad> <good>`
- Have you ever had to try and find where something was fixed?
- `git bisect bad` – no I meant “good”

`git bisect`

- `git bisect start <bad> <good>`
- Have you ever had to try and find where something was fixed?
- `git bisect bad` – no I meant “good”

EVIL!

`git checkout`

- With a file argument it's dangerous but doesn't need a `-f`

`git checkout`

- With a file argument it's dangerous but doesn't need a `-f`
- It wraps two completely separate operations into one command

`git checkout`

- With a file argument it's dangerous but doesn't need a `-f`
- It wraps two completely separate operations into one command
- `checkout -p` messes with your mind

git checkout -p

```
# To remove ’+’ lines, make them ’ ’ lines (context).
# To remove ’-’ lines, delete them.
# Lines starting with # will be removed.
#
# If the patch applies cleanly, the edited hunk will
# immediately be marked for discarding. If it does
# not apply cleanly, you will be given an
# opportunity to edit again. If all lines of the hunk
# are removed, then the edit is aborted and the hunk
# is left unchanged.
```

`git checkout`

- With a file argument it's dangerous but doesn't need a `-f`
- It wraps two completely separate operations into one command
- `checkout -p` messes with your mind

`git checkout`

- With a file argument it's dangerous but doesn't need a `-f`
- It wraps two completely separate operations into one command
- `checkout -p` messes with your mind

EVIL!

`git clone`

- Depending on the transport, it may propogate corruption without noticing
- Inappropriate use of `--mirror` can lose your work

`git clone`

- Depending on the transport, it may propogate corruption without noticing
- Inappropriate use of `--mirror` can lose your work

EVIL!

git commit

- Used to both make normal commits and to “finish off” merges that have needed manual resolving

`git commit`

- Used to both make normal commits and to “finish off” merges that have needed manual resolving

EVIL!

`git pull`

- Often taught as the opposite of `push`

`git pull`

- Often taught as the opposite of `push`
- Does “weird” stuff if you supply too many arguments

`git pull`

- Often taught as the opposite of `push`
- Does “weird” stuff if you supply too many arguments

EVIL!

git push

- Does “weird” stuff if you supply too few arguments

`git push`

- Does “weird” stuff if you supply too few arguments
- It pushes too much by default

git push

- Does “weird” stuff if you supply too few arguments
- It pushes too much by default… unless you’re using newer Git

`git push`

- Does “weird” stuff if you supply too few arguments
- It pushes too much by default… unless you’re using newer Git

EVIL!

`git log`

- By default, presents a linear list of commits regardless of parentage

git log

- By default, presents a linear list of commits regardless of parentage
- Filtering by path removes crucial branch points and merges from the history list

git log

- By default, presents a linear list of commits regardless of parentage
- Filtering by path removes crucial branch points and merges from the history list

EVIL!

git merge

- For most people it creates parents the wrong way around

`git merge`

- For most people it creates parents the wrong way around

EVIL!

git rebase

EVIL!

git rebase

EVIL! but I like it!

`git reset`

- It's too easy to automatically append `--hard`

`git reset`

- It's too easy to automatically append `--hard`
- It wraps two completely separate operations in one command

`git reset`

- It's too easy to automatically append `--hard`
- It wraps **three** completely separate operations in one command

`git reset`

- It's too easy to automatically append `--hard`
- It wraps **three** completely separate operations in one command
- ... half of which are shared with `checkout`

`git reset`

- It's too easy to automatically append `--hard`
- It wraps **three** completely separate operations in one command
- ... half of which are shared with `checkout`

EVIL!

git show

- `git show HEAD`

git show

- `git show HEAD`
- `git show HEAD:file`

git show

- `git show HEAD`
- `git show HEAD:file`
- `git show :file`

git show

- git show HEAD
- git show HEAD:file
- git show :file
- git show :./file

git show

- `git show HEAD`
- `git show HEAD:file`
- `git show :file`
- `git show :./file`
- `git show file`

git show

- `git show HEAD`
- `git show HEAD:file`
- `git show :file`
- `git show :./file`
- `git show file`

EVIL!

git status

- It takes so long on my clone!

git status

- It takes so long on my clone!
- `lstat` every file in my working tree

`git status`

- It takes so long on my clone!
- `lstat` every file in my working tree

EVIL!

git tag

- Creates two totally different types of tag based on whether one of three separate options are used

git tag

- Creates two totally different types of tag based on whether one of **five** separate options are used

`git tag`

- Creates two totally different types of tag based on whether one of **five** separate options are used

EVIL!

Non-evil Git commands

Non-evil Git commands

- `branch`

Non-evil Git commands

- `branch`
- `diff`

Non-evil Git commands

- `branch`
- `diff`
- `fetch`

Non-evil Git commands

- `branch`
- `diff`
- `fetch`
- `grep`

Non-evil Git commands

- `branch`
- `diff`
- `fetch`
- `grep`
- `init`

Non-evil Git commands

- `branch`
- `diff`
- `fetch`
- `grep`
- `init`
- `mv` and `rm`

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

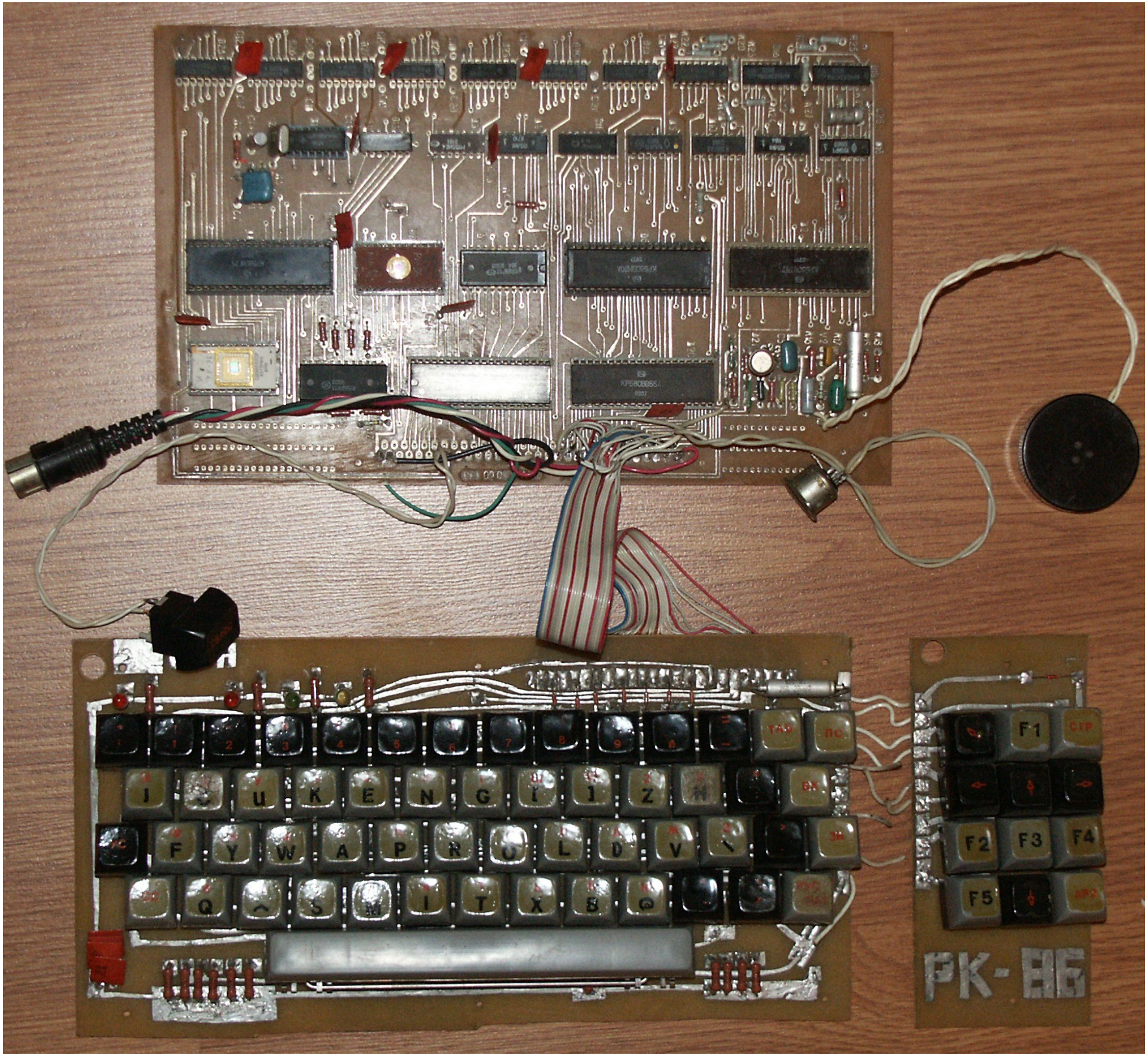
Anders Schau Knatten – OMG Science!

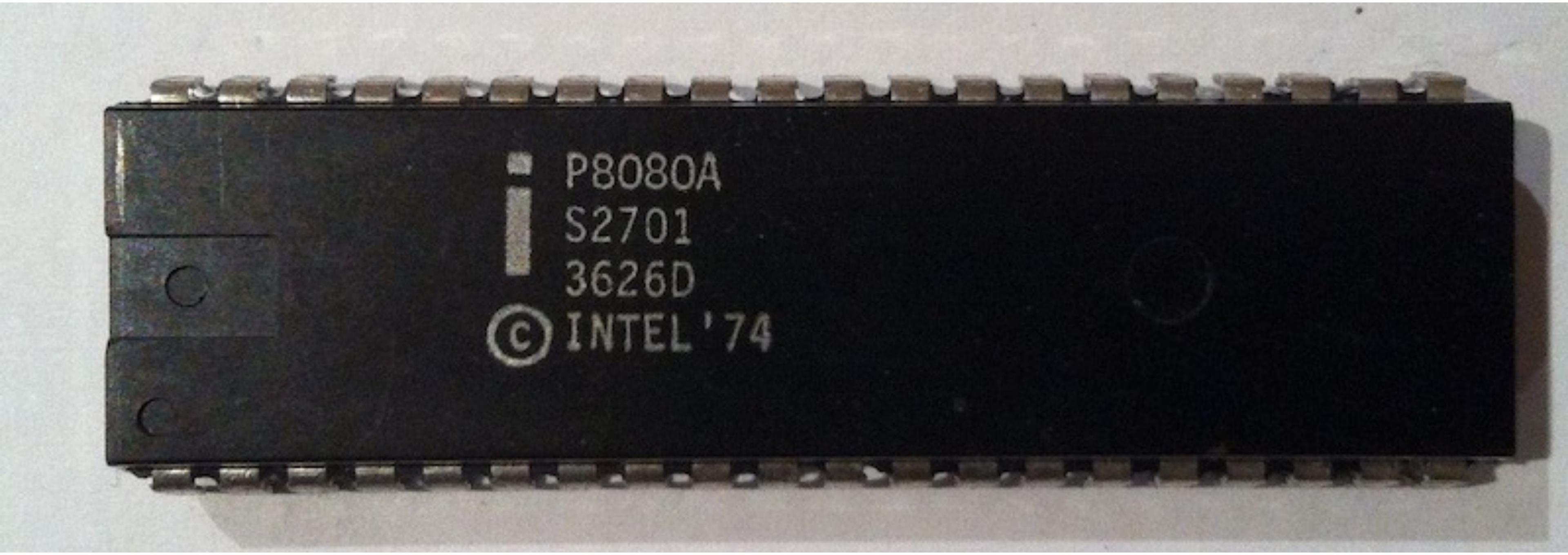
Intel 8080 strikes back

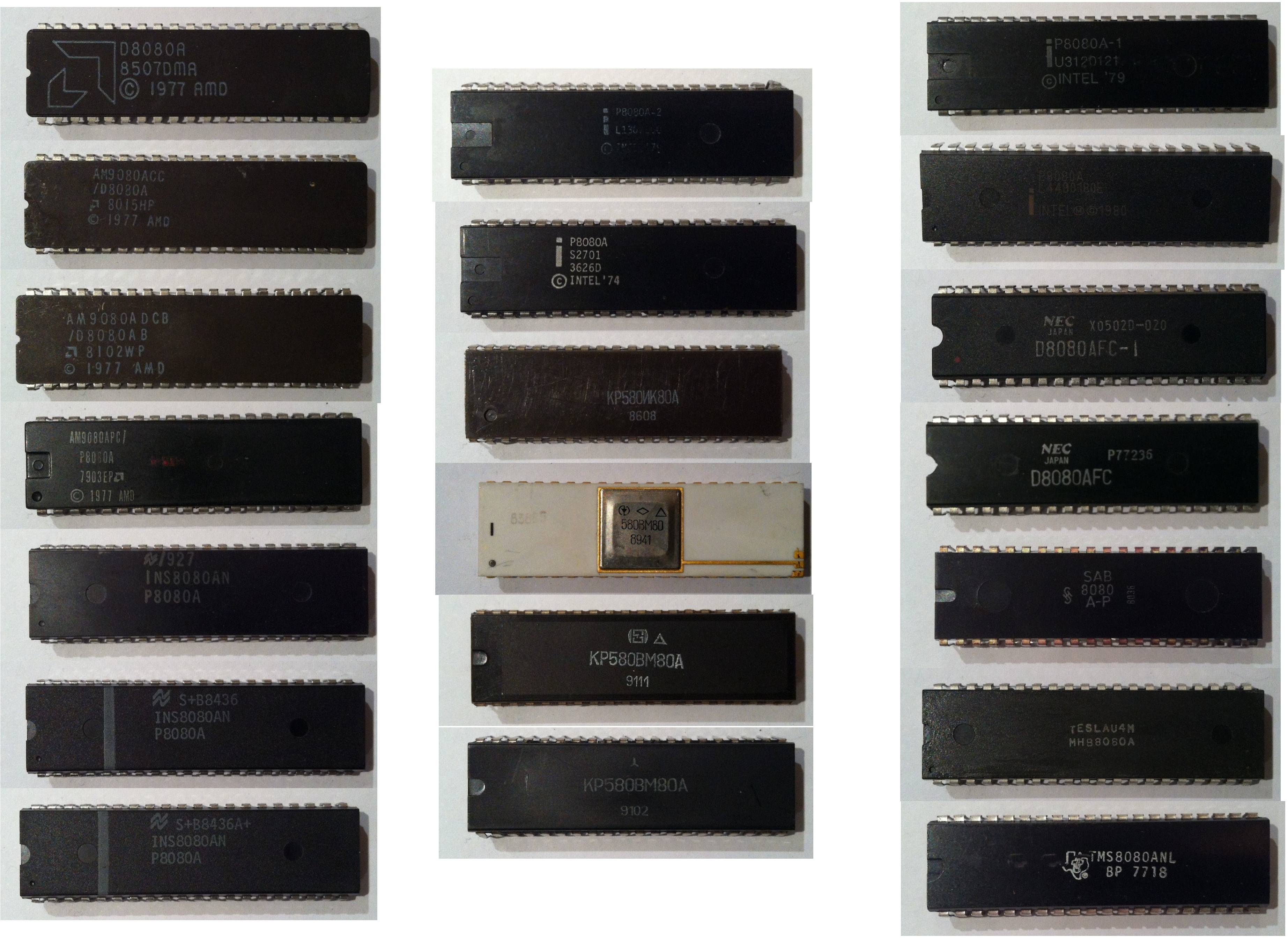
Alexander Demin

http://demin.ws/english

http://github.com/begoon


F1
F2
F3
F4
F5
РК-86


P8080A
S2701
3626D
© INTEL'74


D8080A
8507DMA
© 1977 AMD
AM9080ACC
/D8080A
8015HP
© 1977 AMD
AM9080ADCB
/D8080AB
8102WP
© 1977 AMD
AM9080APC/
P8080A
7903EP
© 1977 AMD
/927
INS8080AN
P8080A
S+B8436
INS8080AN
P8080A
S+B8436A+
INS8080AN
P8080A
P8080A-2
L130
© INTEL '79
P8080A
S2701
3626D
© INTEL '74
КР580ИК80А
8608
580ВМ80
8941
КР580ВМ80А
9111
КР580ВМ80А
9102
P8080A-1
U3120121
© INTEL '79
P8080A
L4490180E
INTEL © 1980
NEC
JAPAN
X0502D-020
D8080AFC-1
NEC
JAPAN
P77236
D8080AFC
SAB
8080
A-P
8036
TESLAU4M
MHB8080A
TMS8080ANL
BP 7718

Options
ACT
PWR
FDX
LNK
OK

Intel 8080 emulators

- Intel 8080 core in JavaScript -- https://github.com/begoon/i8080-js
- Intel 8080 core in C -- https://github.com/begoon/i8080-core
- Radio-86RK online – http://rk86.ru

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Programmers Against Fluffiness

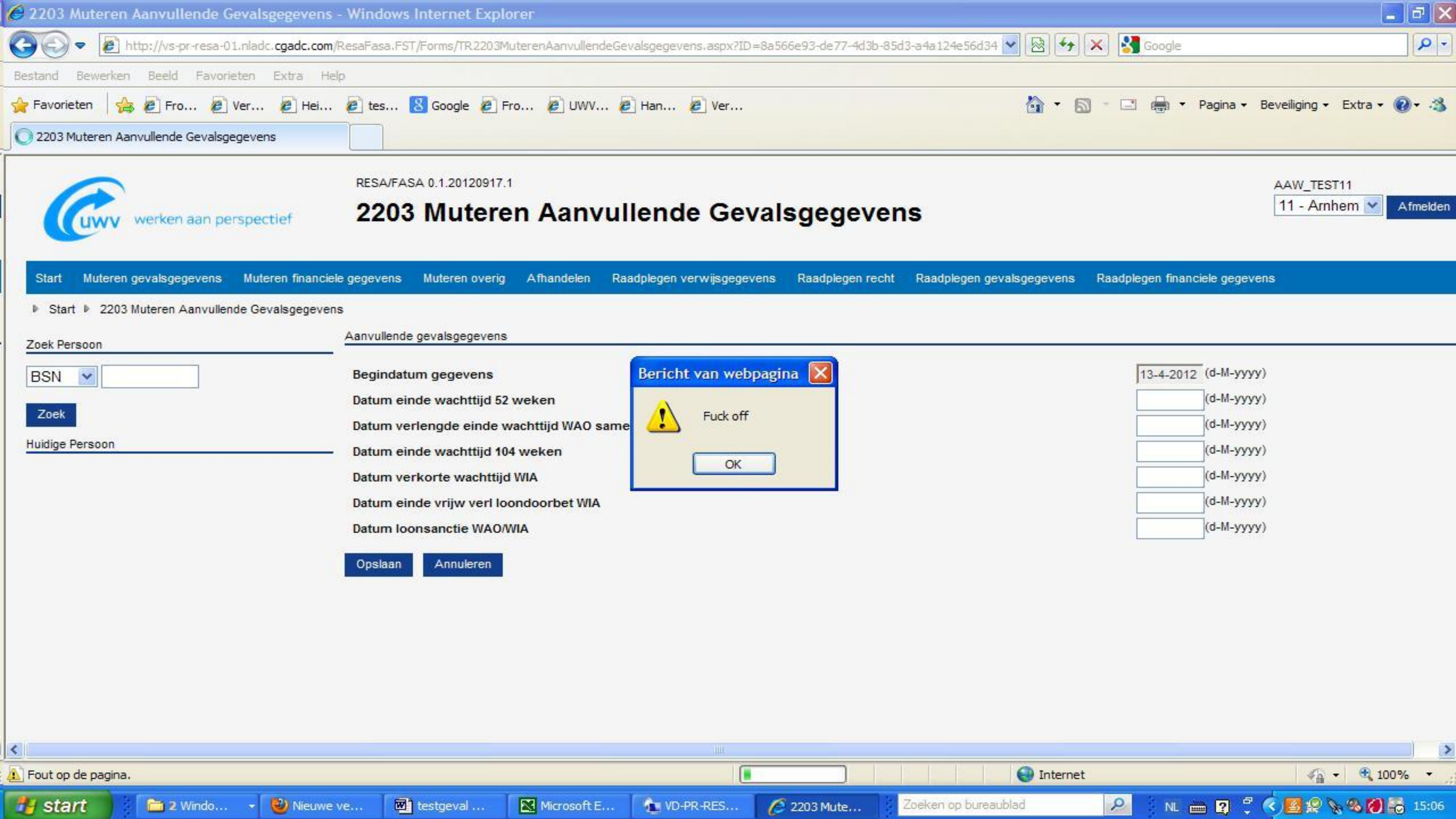
Sander Hoogendoorn
@aahoogendoorn

commodore 64
INTERNET
EMAIL
FILES
VOLUME ▲
VOLUME ▼

Sander Hoogendoorn

Sander Hoogendoorn

On being a developer …



2203 Muteren Aanvullende Gevalsgegevens - Windows Internet Explorer
http://vs-pr-resa-01.nladc.cgadc.com/ResaFasa.FST/Forms/TR2203MuterenAanvullendeGevalsgegevens.aspx?ID=8a566e93-de77-4d3b-85d3-a4a124e56d34
Google
Bestand
Bewerken
Beeld
Favorieten
Extra
Help
Favorieten
Fro…
Ver…
Hei…
tes…
Google
Fro…
UWV…
Han…
Ver…
Pagina
Beveiliging
Extra
2203 Muteren Aanvullende Gevalsgegevens
uwv werken aan perspectief
Afmelden
Start
Muteren gevalsgegevens
Muteren financiele gegevens
Muteren overig
Afhandelen
Raadplegen verwijsgegevens
Raadplegen recht
Raadplegen gevalsgegevens
Raadplegen financiele gegevens
Zoek
Opslaan
Annuleren
Internet
100%
start
2 Windo…
Nieuwe ve…
testgeval …
Microsoft E…
VD-PR-RES…
2203 Mute…
Zoeken op bureaublad
NL
15:06

RESA/FASA 0.1.20120917.1

2203 Muteren Aanvullende Gevalsgegevens

AAW_TEST11

11 - Arnhem

Start ▸ 2203 Muteren Aanvullende Gevalsgegevens

Zoek Persoon

BSN

Huidige Persoon

Aanvullende gevalsgegevens

Begindatum gegevens	13-4-2012	(d-M-yyyy)
Datum einde wachttijd 52 weken		(d-M-yyyy)
Datum verlengde einde wachttijd WAO same		(d-M-yyyy)
Datum einde wachttijd 104 weken		(d-M-yyyy)
Datum verkorte wachttijd WIA		(d-M-yyyy)
Datum einde vrijw verl loondoorbet WIA		(d-M-yyyy)
Datum loonsanctie WAO/WIA		(d-M-yyyy)

Bericht van webpagina

Fuck off

OK

Fout op de pagina.

All’s well
that ends well

Agile

Agile (Un)Conferences...

At the wrong reenactment

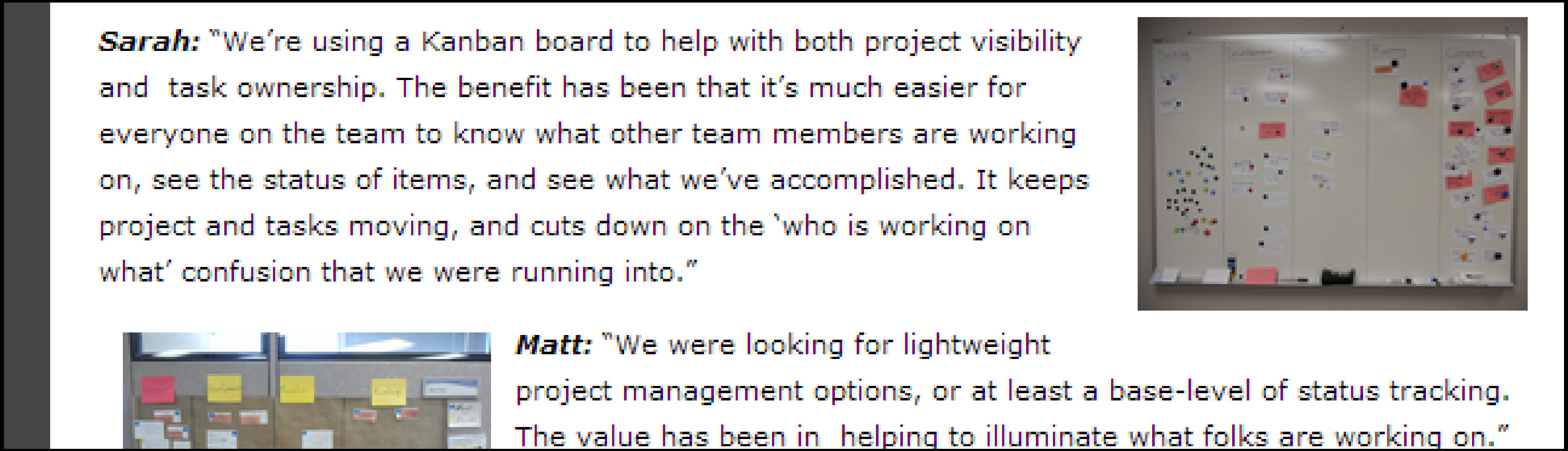
The Agile Suits

The Agile Hippies

About Fences

About Fences

Kindergarden Agile



Sarah: "We're using a Kanban board to help with both project visibility and task ownership. The benefit has been that it's much easier for everyone on the team to know what other team members are working on, see the status of items, and see what we've accomplished. It keeps project and tasks moving, and cuts down on the 'who is working on what' confusion that we were running into."

Matt: "We were looking for lightweight project management options, or at least a base-level of status tracking. The value has been in helping to illuminate what folks are working on."

Kindergarden Agile

“Make sure you don’t miss the agile elephant versus the waterfall elephant in the lobby.”

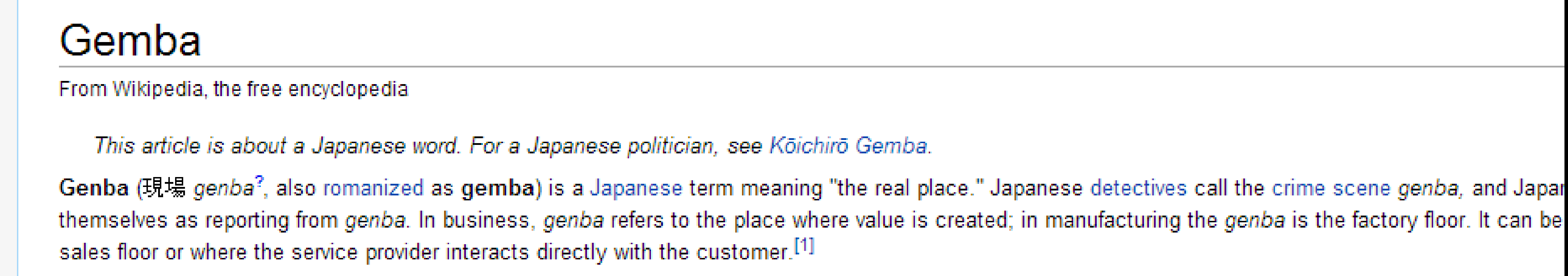
“During this session we are going to discuss the Hapiness Index of projects.”

“Add *Ready for Celebration* before the Done column on your Kanban board”

Open Door Wisdom

“If your retrospectives don’t add value to your project, you should change your retrospectives.”

Japanifying Agile



Gemba

From Wikipedia, the free encyclopedia

This article is about a Japanese word. For a Japanese politician, see Kōichirō Gemba.

Genba (現場 *genba*?, also romanized as **gemba**) is a Japanese term meaning "the real place." Japanese detectives call the crime scene *genba,* and Japar themselves as reporting from *genba*. In business, *genba* refers to the place where value is created; in manufacturing the *genba* is the factory floor. It can be sales floor or where the service provider interacts directly with the customer.[1]

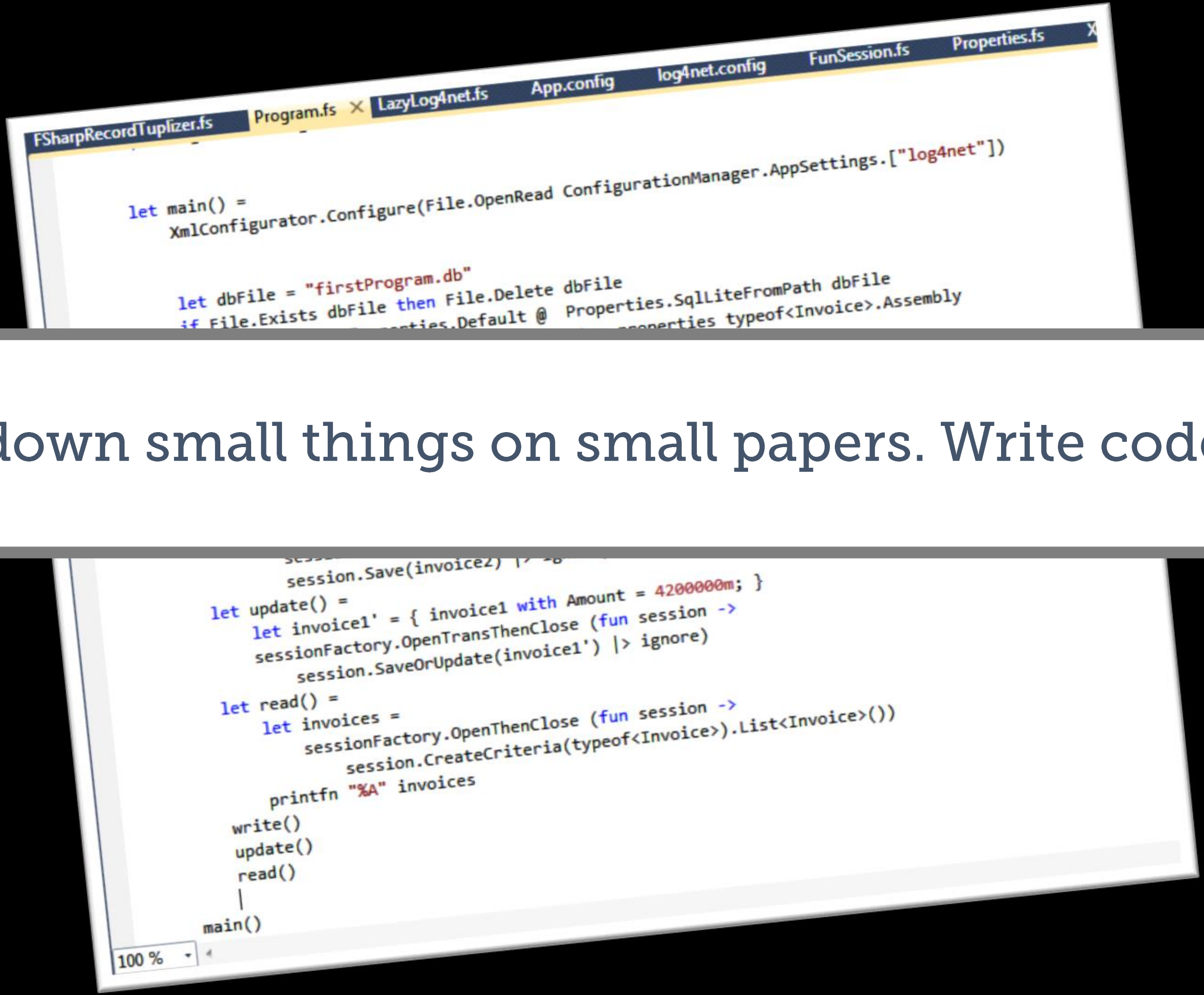
artists against apartheid

programmers against fluffiness

Programmers Against Flufiness

Just write down small things on small papers. It’s your kaizen.

Programmers Against Fluffiness



```
let main() =
    XmlConfigurator.Configure(File.OpenRead ConfigurationManager.AppSettings.["log4net"])

    let dbFile = "firstProgram.db"
    if File.Exists dbFile then File.Delete dbFile
              erties.Default @  Properties.SqlLiteFromPath dbFile
                               roperties typeof<Invoice>.Assembly
```

```
            session.Save(invoice2) |>
    let update() =
        let invoice1' = { invoice1 with Amount = 4200000m; }
        sessionFactory.OpenTransThenClose (fun session ->
            session.SaveOrUpdate(invoice1') |> ignore)
    let read() =
        let invoices =
            sessionFactory.OpenThenClose (fun session ->
                session.CreateCriteria(typeof<Invoice>).List<Invoice>())
        printfn "%A" invoices
    write()
    update()
    read()

main()
```

Don’t just write down small things on small papers. Write code. It’s what we do.

www.sanderhoogendoorn.com

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Emulating N3564

Blocking I/O code

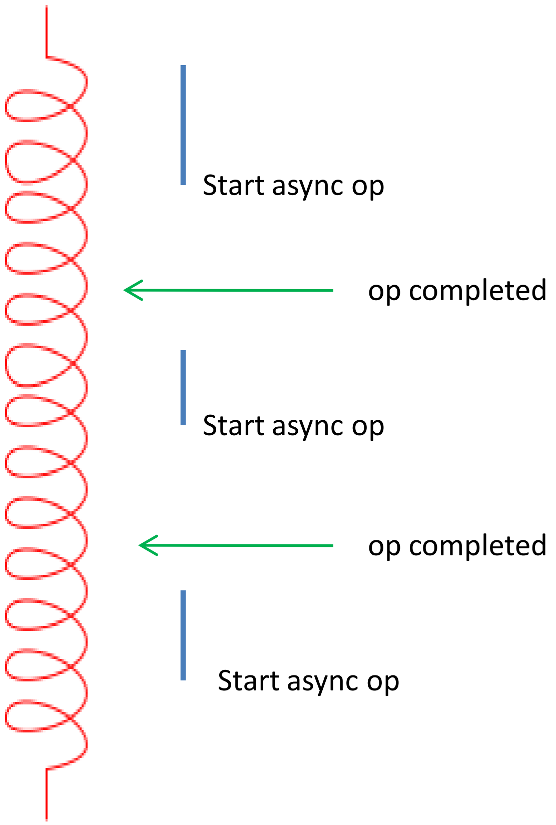
```
void f()
{
  while (!stop) {
    try {
      std::string content = http_get("<url>");

      std::string result = transmogrify(content);

      if (!result.empty())
        write_file("<file_path>", result);
    }
    catch (const io_error&) {}

    std::this_thread::sleep_for(std::chrono::seconds(5));
  }
}
```

Blocking I/O code

```
void f()
{
  while (!stop) {
    try {
      std::string content = http_get("<url>");                  ✗

      std::string result = transmogrify(content);

      if (!result.empty())
        write_file("<file_path>", result);                       ✗
    }
    catch (const io_error&) {}

    std::this_thread::sleep_for(std::chrono::seconds(5));        ✗
  }
}
```

An async version

- Callback-based, asynchronous APIs
- Event loop/message pump
- Chop original code into non-blocking pieces
- A state machine
- Feed the right code bits into the pump as the async operations complete



std::future<T>

```cpp
std::future<std::string> http_get(const std::string& url);

std::future<void> write_file(const std::string& path,
                             const std::string& content);

std::future<void> timeout(std::chrono::seconds duration);
```

std::future<T>

```
std::future<std::string> http_get(const std::string& url);

std::future<void> write_file(const std::string& path,
                             const std::string& content);

std::future<void> timeout(std::chrono::seconds duration);
```

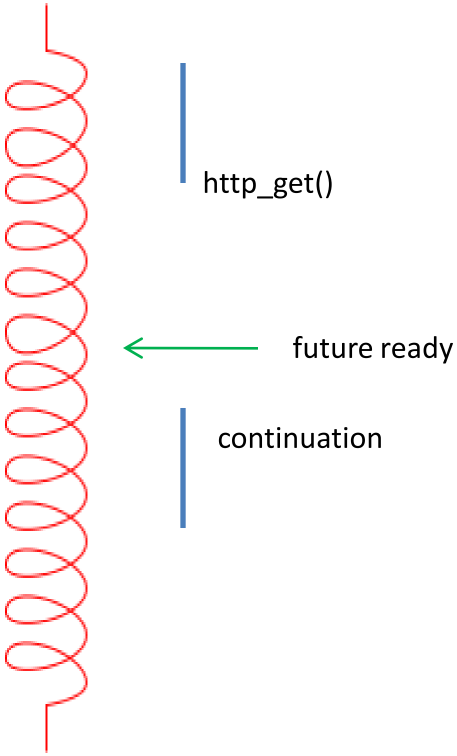
Problem:

```
auto f = http_get("<url>");
std::string content = f.get();
use(content);
```

✘

std::future::then()

```
auto f = http_get("<url>");
f.then([] (std::future<std::string> g) {
  std::string content = g.get();
  use(content);
}
```



http_get()

future ready

continuation

A standardized way of specifying callbacks/errbacks (N3558)

boost::coroutines

```
boost::coroutines::coroutine<void ()>
```

```
void f(wait_context& ctx)
{
  // ...
  std::string content = ctx.await(http_get(”<url>”));
  // ...
}
```

boost::coroutines

```
boost::coroutines::coroutine<void ()>

void f(wait_context& ctx)
{
  // ...
  std::string content = ctx.await(http_get(”<url>”));
  // ...
}
```

```
template <typename T>
T wait_context::await(std::future<T>&& f) {
  auto shared_fut = f.share();
  then(shared_fut, [this] (std::shared_future<T>) {
    resume_coroutine();
  });
  coroutine_yield();
  return shared_fut.get();
}
```

Powered by boost::coroutines

```cpp
void f(resumable<void>::wait_context& ctx)
{
  while (!stop) {
    try {
      std::string content = ctx.await(http_get("<url>"));

      std::string result = transmogrify(content);

      if (!result.empty())
        ctx.await(write_file("<file_path>", result));
    }
    catch (const io_error&) {}

    ctx.await(timeout(std::chrono::seconds(5)));
  }
}
```

Powered by N3564

```
std::future<void> f() resumable
{
  while (!stop) {
    try {
      std::string content = await http_get("<url>");

      std::string result = transmogrify(content);

      if (!result.empty())
        await write_file("<file_path>", result);
    }
    catch (const io_error&) {}

    await timeout(std::chrono::seconds(5));
  }
}
```

Thank you!

No threads were blocked in the making of this presentation.

Stig Sandnes
stigsand@cisco.com

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Passionate vs. Professional

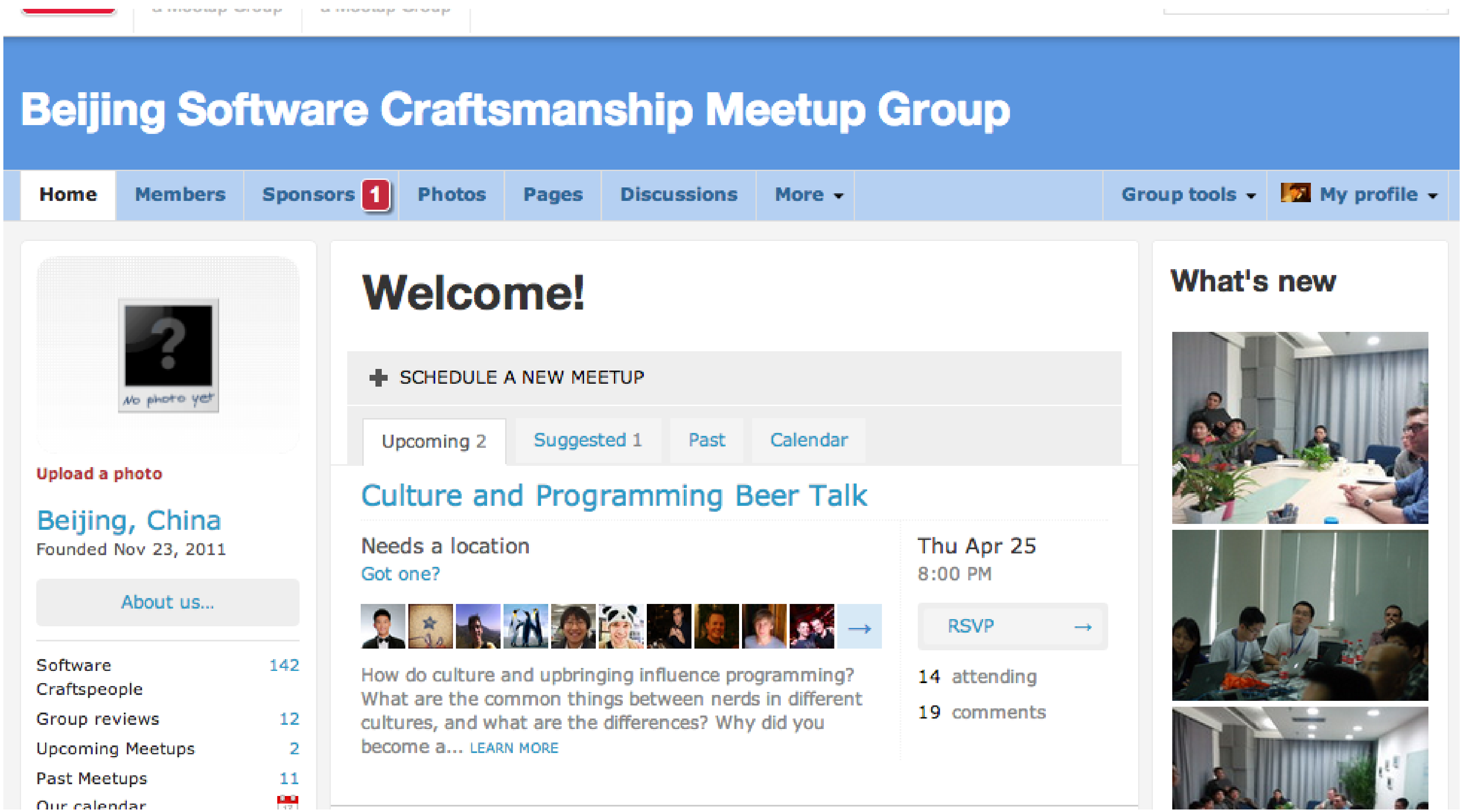
A response to “Why are (only) we here?”

Mike Long

Alistair McKinnell's 3 Rules

- It works
- It is easy to understand
- It is easy to change

Find a community



Beijing Software Craftsmanship Meetup Group

Beijing, China

Founded Nov 23, 2011

Software Craftspeople	142
Group reviews	12
Upcoming Meetups	2
Past Meetups	11
Our calendar	

Welcome!

Culture and Programming Beer Talk

Needs a location

Got one?

Thu Apr 25

8:00 PM

14 attending

19 comments

How do culture and upbringing influence programming? What are the common things between nerds in different cultures, and what are the differences? Why did you become a... LEARN MORE

What's new

accu 2013 Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Testing libc++ with *Sanitizer

Marshall Clow
marshall@idio.com

What is *Sanitizer?

- Attempt to give the facilities of valgrind at a much lower cost
- Compiler pass + custom runtime
- Several different combinations

libc++ tests

- ~4850 tests
- takes about 30 minutes to run
- Baseline (Mac OS X): 12 failures

Address Sanitizer

- http://clang.llvm.org/docs/AddressSanitizer.html
- Tests for:
  - Out of bounds reads/writes
  - Use after free
  - Use after return
  - Double Free

Initial results

- Took 92 minutes to run the tests (3x slowdown)
- 54 failures (instead of 12)

What failed?

- 11 tests failed with a out-of-bounds write to the heap
- 2 tests failed with out-of-bounds read
- 25 tests failed with to load with a missing symbol.
- 4 tests failed with memory allocation errors
- 12 tests failed that we started with

UBSan

- Checks for undefined behavior
- http://clang.llvm.org/docs/UsersManual.html#controlling-code-generation

Full writeups

- http://blog.llvm.org/2013/03/testing-libc-with-address-sanitizer.html
- http://blog.llvm.org/2013/04/testing-libc-with-fsanitizeundefined.html

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

An attempt to add ref-qualifiers to assignment operators in the C++ Standard, Frankfurt, 2009

Niels Dekker

NielsDekker@xs4all.nl

LKEB, Leiden University Medical Center

Pix by John Lakos and Jens Maurer

Lvalue/rvalue ref-qualifiers

Ref-qualifiers specify whether a *member function* can be used for an lvalue, or an rvalue *(Bronek Kozicki, Daveed Vandevoorde).*

only for an *lvalue*

```
class Foo
{
public:
  void SetName(string) &;

  string& GetName() &;
  string&& GetName() &&;
  const string& GetName() const &;
};
```

Assignment to an rvalue???

Forbidden for built-in types (int, float, pointers, etc.):

```
int i;
int f();
```

```
f() = i;
```

Error: assignment to an rvalue!

Assignment to an rvalue???

Allowed for Standard Library types!

```
std::complex<double> c;
std::complex<double> f();

f() = c;
```

Compiles, however silly!

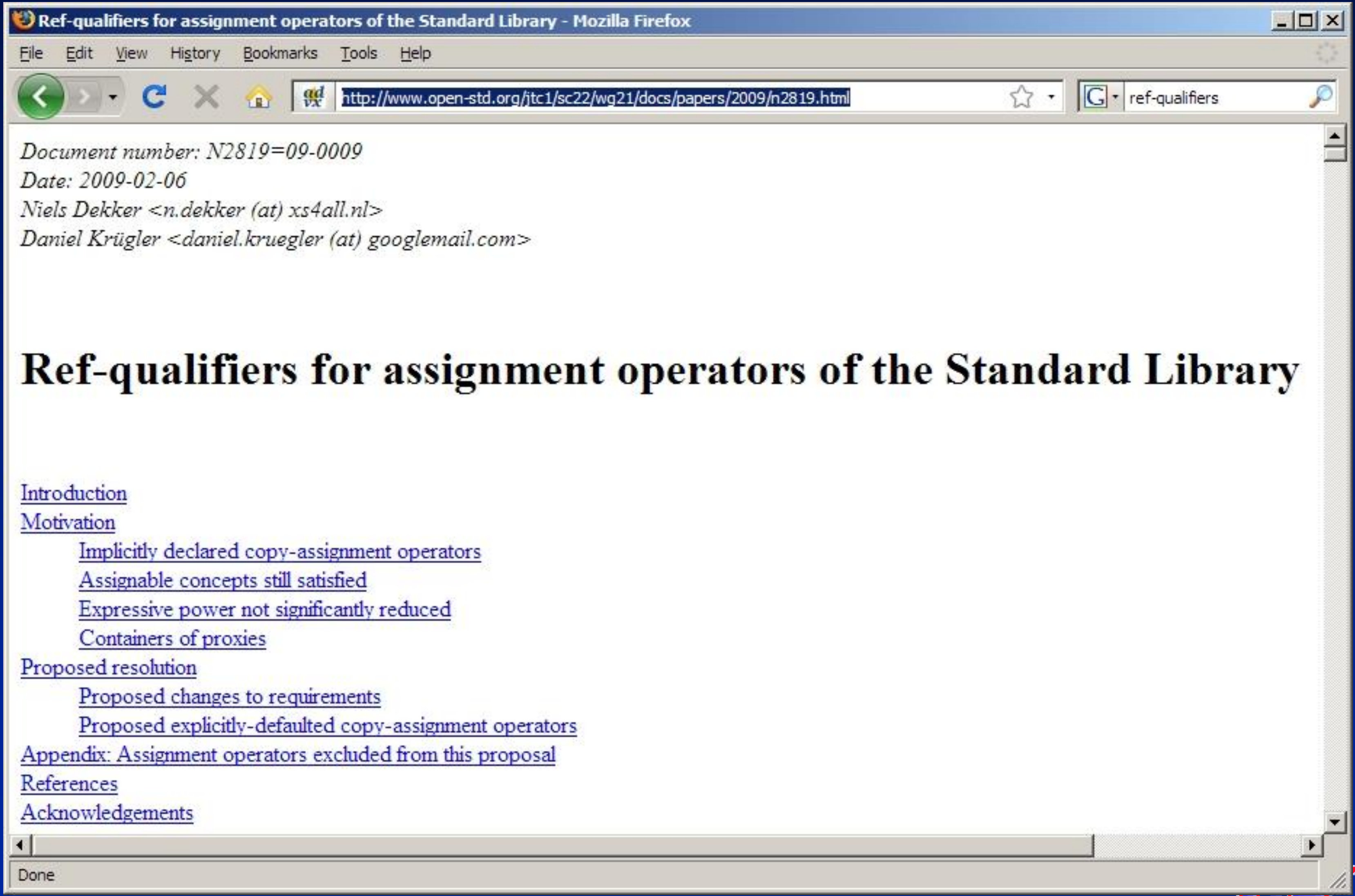
Assignment to an rvalue… typical bug

Assigment operator looks so much like equality operator!

```
std::shared_ptr<int> f();


if ( f() = nullptr )
{

}
```

Bug!

The paper by me and Daniel Krügler



Ref-qualifiers for assignment operators of the Standard Library - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2009/n2819.html
ref-qualifiers
Done

Document number: N2819=09-0009
Date: 2009-02-06
Niels Dekker <n.dekker (at) xs4all.nl>
Daniel Krügler <daniel.kruegler (at) googlemail.com>

Ref-qualifiers for assignment operators of the Standard Library

Ref-qualifiers for assignment operators of the Standard Library

```
string& operator=(const string&) &
```

- Add an lvalue ref-qualifier to over 200 assignment operators, including std::vector. std::string, std::shared_ptr, etc.
- Add 150 explicitly-defaulted assignment operators

Monday at Library subgroup 1

- Presentation and discussion (30 minutes)
- Positive feedback
- Encouraged Core Working Group to look at a related proposal : **Where possible, add a ref qualifier to an implicitly declared assignment operator** *(Alberto Ganesh Barbati)*.
- Awaited feedback from Core Working Group

Tuesday at the Core working group

Ref qualifiers for implicitly declared assignment operators?

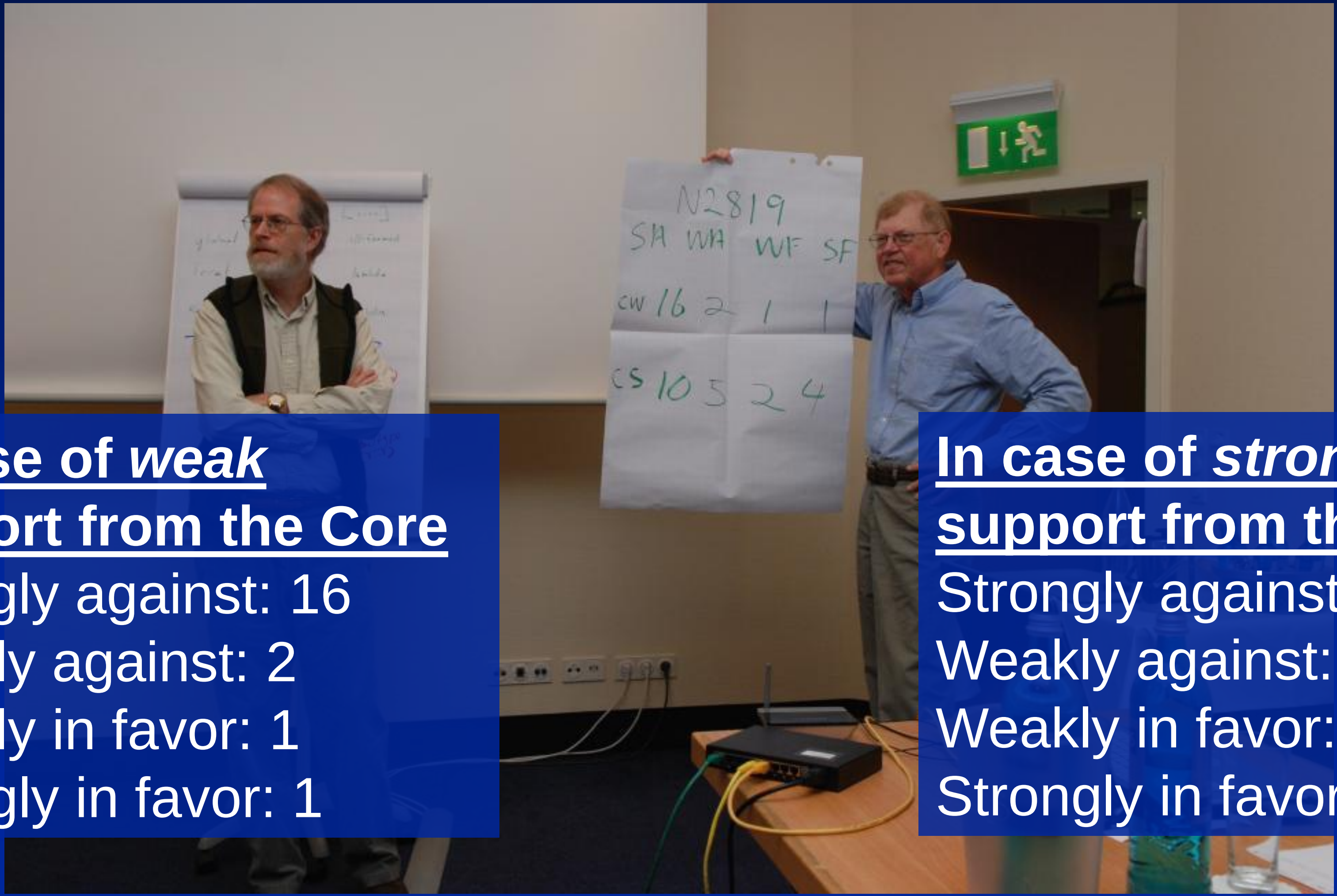
Tuesday at the Core working group

Thursday back at the Library working group

Another presentation of mine, but...

- People felt unfamiliar with ref-qualifiers
- Is there enough demand from users?
- Breaks old source code
- Breaks ABI of the Standard C++ Library
- Too late for C++0x
- Too many changes (over 200 + 150)

Outcome of voting from LWG



In case of *weak* support from the Core

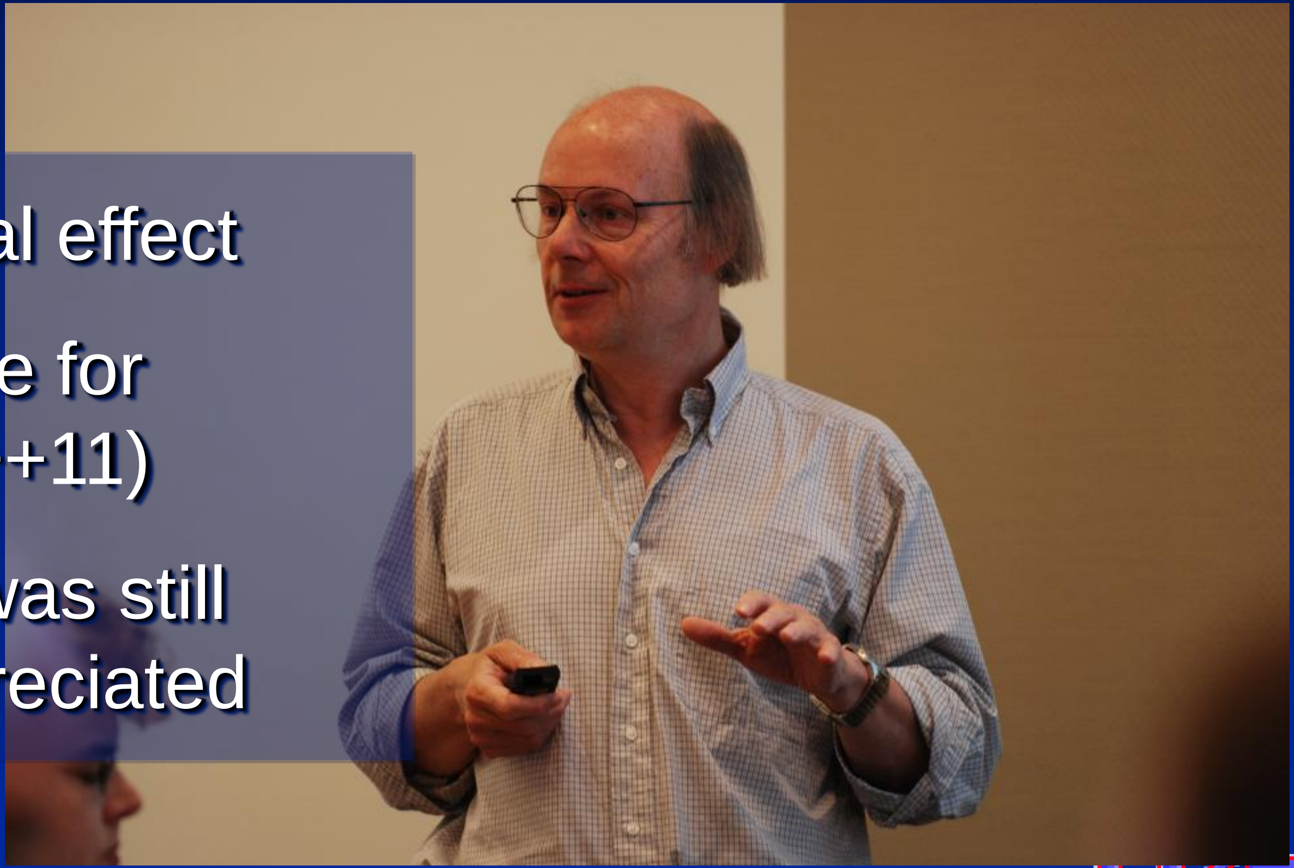
Strongly against: 16
Weakly against: 2
Weakly in favor: 1
Strongly in favor: 1

In case of *strong* support from the Core

Strongly against: 10
Weakly against: 5
Weakly in favor: 2
Strongly in favor: 4

Thursday afternoon...

Bjarne opened a session of the Evolution Working Group on this subject.



- Fear of viral effect
- Not feasible for C++0x (C++11)
- Our work was still highly appreciated

Conclusion

- For me personally: a great experience
- Who knows… proposal might be revisited when the time is right…

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Variables with no Name ;^(

A sad story of thousands of unnamed variables who go out of scope every day

```
public void Wibble()
{
    using (SPSite site = new SPSite("http://portal"))
    {
        using(SPWeb web = site.OpenWeb())
        {
            DoIt(web);
        }
    }
}
```

These variables have no name!

```
public void Wibble()
{
    using (SPSite site = new SPSite("http://portal"))
    {
        using(SPWeb web = site.OpenWeb())
        {
            DoIt(web);
        }
    }
}
```

I don’t mean…

```
new Wibble(p1, p2);
```

A small digression…

Where do you get these from?

Where does this come from?

```
protected CString m_strDescription;
```

Break it into pieces:

```
m_strDescription;

Scope = m_
Type = str
Name = Description
```

The type is the name!

```
public void Wibble()
{
    using (SPSite site = new SPSite("http://portal"))
    {
        using(SPWeb web = site.OpenWeb())
        {
            DoIt(web);
        }
    }
}
```

Good for Copy & Paste, though!

```
public void Wibble()
{
    using (SPSite site = new SPSite("http://portal"))
    {
        using(SPWeb web = site.OpenWeb())
        {
            DoSomethingElse(web);
        }
    }
}
```

Other Offenders

```
for(int i = 0; i > count; ++i(){…}
```

Let’s give them a name

```
public void Wibble()
{
    using (SPSite partner = new SPSite("http://portal"))
    {
        using(SPWeb dossier = site.OpenWeb())
        {
            DoIt(dossier);
        }
    }
}
```

The Other Offenders

```
for(int i = 0; i > count; ++i){…}
```

OK, that’s cannonical

```
Paragraph findHit = new Paragraph();
```

Please the thoughtful

Give your variables a name!

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

C++ EXTENSION METHODS

@phil_nash

```
class A:

  foo()
  bar()

  data
```

```
class A:

  foo()
  bar()

  data
```

```
+ explode()
```

DYNAMIC LANGUAGES

```
setattr(cls, func.__name__,
        types.MethodType(func, cls))
```

```
setattr(cls, func.__name__,
        types.MethodType(func, cls))
```

```
var a = {};
a.f1 = function(){};
```

```
setattr(cls, func.__name__,
        types.MethodType(func, cls))
```

```
var a = {};
a.f1 = function(){};
```

```
struct objc_method myMethod;
myMethod.method_name = sel_registerName("sayHello");
myMethod.method_imp  = sayHello;

struct objc_method_list * myMethodList;
myMethodList = malloc (sizeof(struct objc_method_list));
myMethodList->method_count = 1;
myMethodList->method_list[0] = myMethod;

class_addMethods ( [EmptyClass class], myMethodList );
```

Static Languages

```
namespace ExtensionMethods {
    public static class MyExtensions {
        public static int WordCount(this String str) {
            return str.Split(new char[] { ' ', '.', '?' },
                             StringSplitOptions.RemoveEmptyEntries).Length;
        }
    }
}
```

```
namespace ExtensionMethods {
    public static class MyExtensions {
        public static int WordCount(this String str) {
            return str.Split(new char[] { ' ', '.', '?' },
                             StringSplitOptions.RemoveEmptyEntries).Length;
        }
    }
}
```

```
@implementation NSString (reverse)

-(NSString *) reverseString {
  NSMutableString *reversedStr;
  int len = [self length];
  reversedStr = [NSMutableString stringWithCapacity:len];
  while (len > 0)
    [reversedStr appendString:
         [NSString stringWithFormat:@"%C", [self characterAtIndex:--len]]];
  return reversedStr;
}
```

C++

C++

THE INTERFACE PRINCIPLE

```
namespace NS {
    class T { };
    void f( T );
}

int main() {
    NS::T object;
    f(object);
}
```

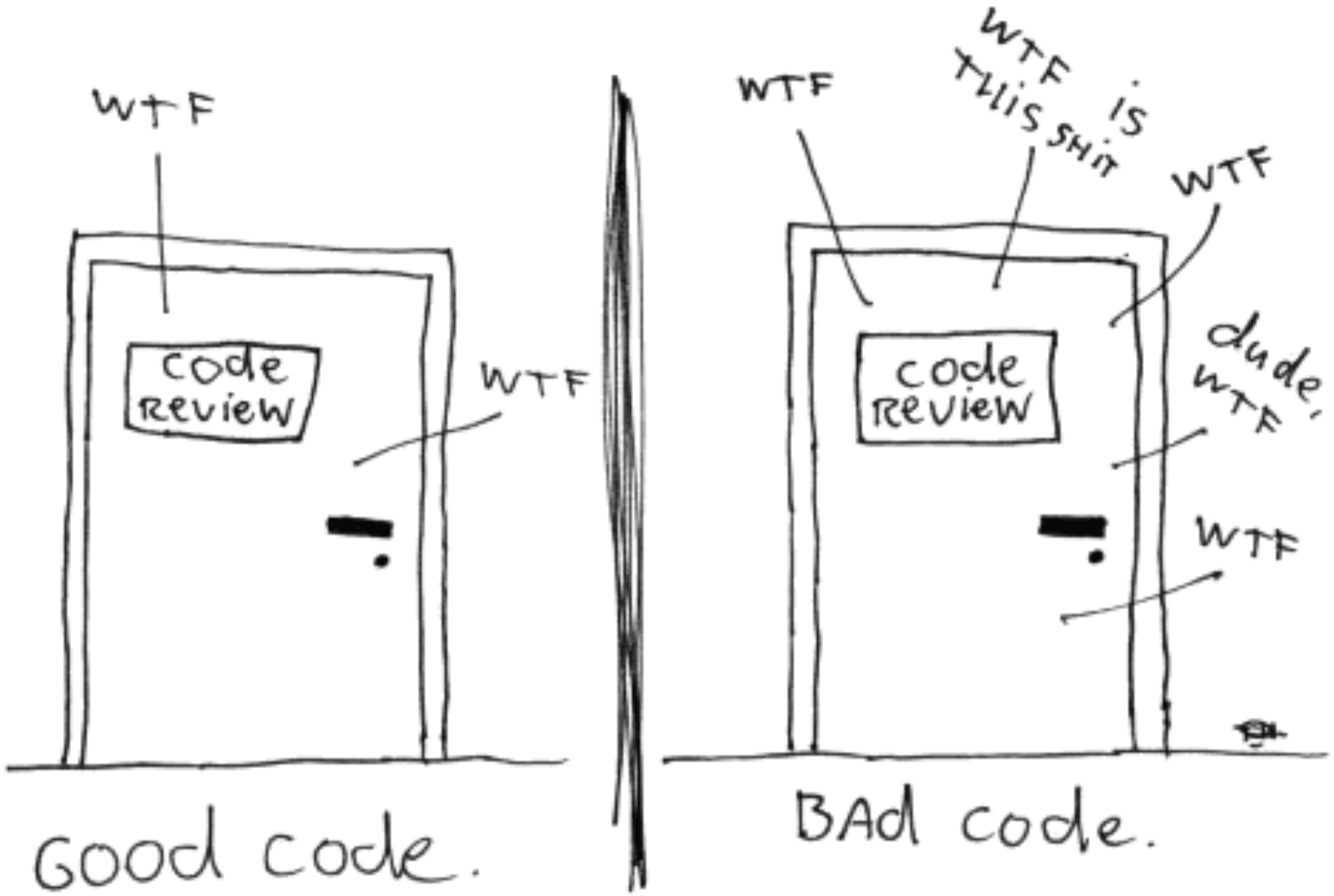
```
namespace NS {
    class T { };
    void f( T );
}

int main() {
    NS::T object;
    object.f();
}
```

```
namespace NS {
    class T { };
    void f( T );
}

int main() {
    NS::T* object = new NS::T();
    object->f();
}
```

```
object->f()
```

```
object<-f()
```

The only valid measurement of code quality: WTFs/minute



(c) 2008 Focus Shift

```
struct Widget : Extendible<Widget> {
    Widget( int size, int weight ) : size( size ), weight( weight ) {}

    int size;
    int weight;
};
```

```
struct Widget : Extendible<Widget> {
    Widget( int size, int weight ) : size( size ), weight( weight ) {}

    int size;
    int weight;
};
```

```
struct print : ExtMethod<print> {
    void operator()( Widget& w ) {
        std::cout << "size: " << w.size << ", weight: " << w.weight << std::endl;
    }
};
```

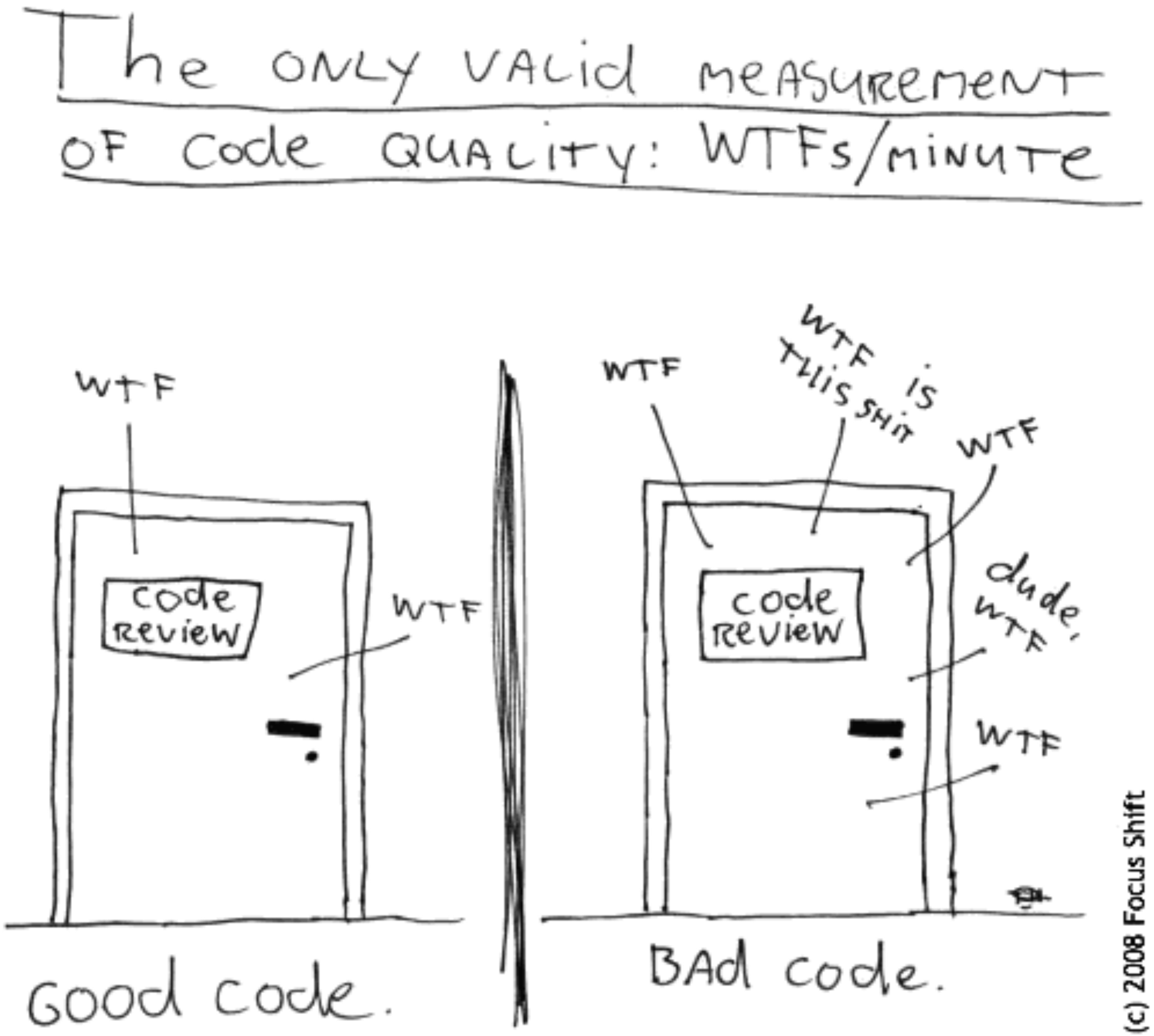
```
Widget w( 4, 10 );
w<-print();
```

```
struct Widget : Extendible<Widget> {
    Widget( int size, int weight ) : size( size ), weight( weight ) {}

    int size;
    int weight;
};
```

```
struct print : ExtMethod<print> {
    void operator()( Widget& w ) {
        std::cout << "size: " << w.size << ", weight: " << w.weight << std::endl;
    }
};
```

```
struct density : ExtMethod<density, float> {
    float operator()( Widget& w ) {
       return (float)(w.weight / w.size);
    }
};
```

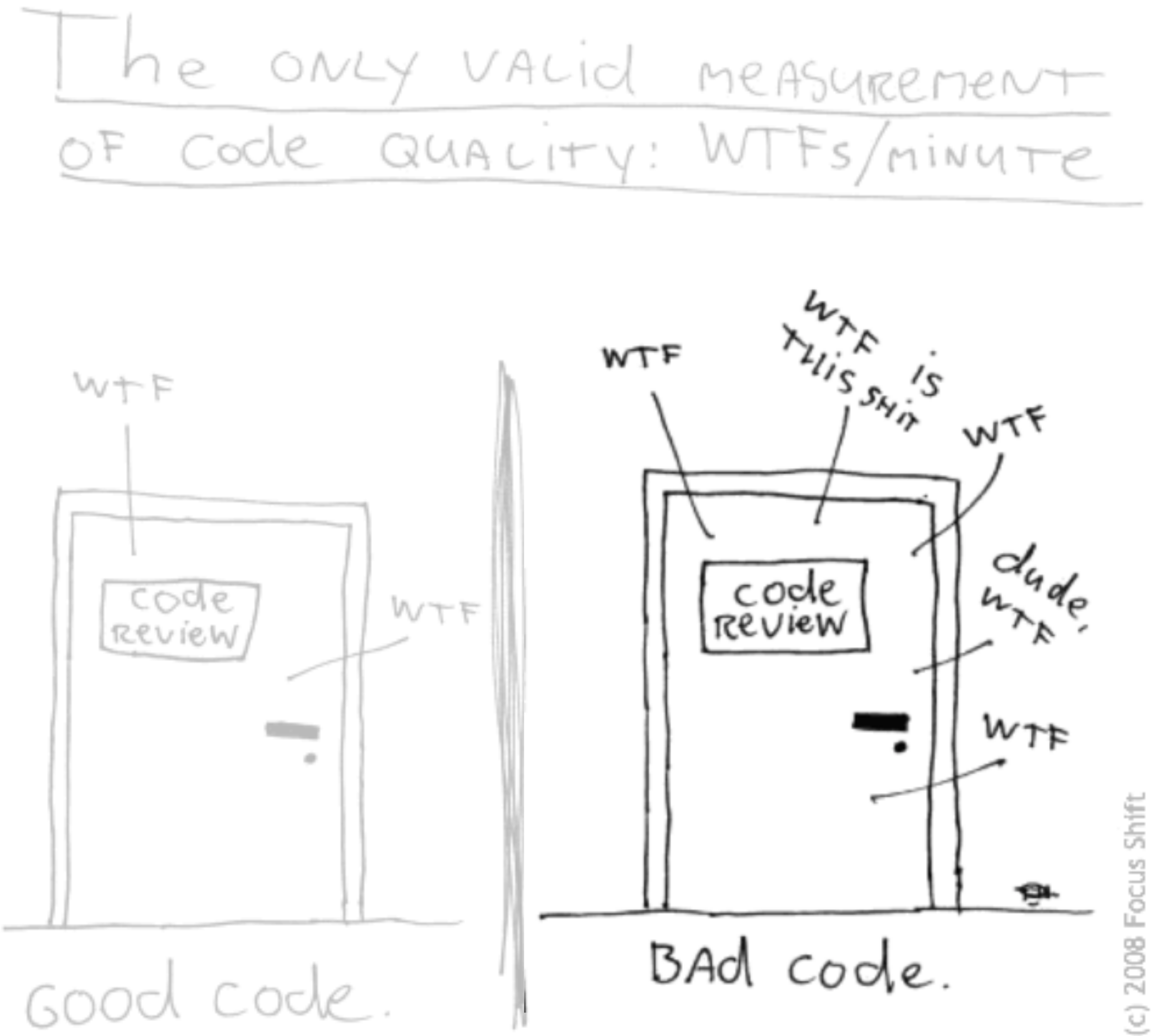
```
Widget w( 4, 10 );
w<-print();

float d = w<-density();
```



(c) 2008 Focus Shift

```
template<typename T, typename ReturnT=void>
struct ExtMethod {
    ExtMethod& operator - () {
        return *this;
    }
    template<typename U>
    ReturnT operator()( U& obj ) {
        return static_cast<T*>(this)->operator()( obj );
    }
};
```

```
template<typename Derived>
struct Extendible
{
    template<typename T, typename ReturnT>
    ReturnT operator < ( ExtMethod<T, ReturnT>& extMethod ) {
        return extMethod( static_cast<Derived&>( *this ) );
    }
};
```


The only valid measurement of code quality: WTFs/minute
WTF
code review
WTF
Good code.
WTF
WTF is this shit
WTF
code review
dude, WTF
WTF
BAd code.
(c) 2008 Focus Shift

```
template<typename T, typename ReturnT=void>
struct ExtMethod {
    ExtMethod& operator - () {
        return *this;
    }
    template<typename U>
    ReturnT operator()( U& obj ) {
        return static_cast<T*>(this)->operator()( obj );
    }
};
```

```
template<typename Derived>
struct Extendible
{
    template<typename T, typename ReturnT>
    ReturnT operator < ( ExtMethod<T, ReturnT>& extMethod ) {
        return extMethod( static_cast<Derived&>( *this ) );
    }
};
```

```
template<typename T, typename ReturnT=void>
struct ExtMethod {
    ExtMethod& operator - () {
        return *this;
    }
    template<typename U>
    ReturnT operator()( U& obj ) {
        return static_cast<T*>(th                     );
    }
};
```

Please don’t do this in production code!

```
templat
st


          ypename T, typename ReturnT>
      T operator < ( ExtMethod<T, ReturnT>& extMethod ) {
        return extMethod( static_cast<Derived&>( *this ) );
    }
};
```

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!

Who Needs C++ When You Have D and Go?

Russel Winder

email: russel@winder.org.uk
xmpp: russel@winder.org.uk
twitter: @russel_winder
Web: http://www.russel.org.uk

controversy

noun

dispute, argument, or debate, especially one concerning a matter about which there is strong disagreement

Calculate the sum of the squares of the numbers between 0 and 100 that are divisible by 7.

```
int sequential_iterative() {
  auto sum = 0;
  foreach (i; 0 .. 100) {
    if (i % 7 == 0) {
      sum += i * i;
    }
  }
  return sum;
}
```

```
func sequential_iterative() (sum int) {
    for i := 0; i < 100; i++ {
        if (i % 7 == 0) {
            sum += i * i
        }
    }
    return
}
```

```
int sequential_iterative() {
  auto sum = 0;
  for (auto i = 0; i < 100; ++i) {
    if (i % 7 == 0) {
      sum += i * i;
    }
  }
  return sum;
}
```

So all native code programming languages are the same?

We are, of course, expected
to be more declarative these days…

Possibly even functional…

```
reduce!"a + b"(map!"a * a"(filter!"a % 7 == 0"(iota(100))))
```

```
functools.Inject(func (a, b int) int { return a + b },
        functools.Collect(func (i int) int { return i * i },
            functools.Filter(func (i int) bool { return i % 7 == 0 },
                make([]int, 100))))
```

```
int sequential_declarative() {
  std::vector<int> numbers (100);
  std::iota(numbers.begin(), numbers.end(), 1);
  std::list<int> filtered;
  std::copy_if(numbers.begin(), numbers.end(),
      std::back_insert_iterator<std::list<int>>(filtered), [=] (int i) { return i % 7 == 0; });
  std::vector<int> squares (filtered.size());
  std::transform(filtered.begin(), filtered.end(),
      std::back_insert_iterator<std::vector<int>>(squares), [=] (int i) { return i * i;});
  return std::accumulate(squares.begin(), squares.end(), 0, [=] (int i, int j) { return i + j;});
}
```

What about being fluent…

```
iota(100).filter!"a % 7 == 0"().map!"a * a"().reduce!"a + b"()
```

```
functools.IntSlice(make([]int, 100)).
    Filter(func (i int) bool { return i % 7 == 0 }).
        Collect(func (i int) int { return i * i }).
            Inject(func (a, b int) int { return a + b })
```

?

Execute it!

Use D

Use Go

Use Python

Anything but C++

http://www.dlang.org

http://www.go-lang.org

Who Needs C++ When You Have D and Go?

Russel Winder

email: russel@winder.org.uk
xmpp: russel@winder.org.uk
twitter: @russel_winder
Web: http://www.russel.org.uk

accu 2013

Lightning Talks

Dmitry Kandalov – IDE plugin in 5 minutes (for real)

Charles Bailey – Is Git Evil?

Alexander Demin – Intel 8080 Strikes Back

Sander Hoogendoorn – Programmers Against Fluffiness

Stig Sandnes – Emulating N3564

Mike Long – Passionate vs. Professional

Marshall Clow – Testing libc++ with *Sanitizer

Niels Dekker – An attempt to add ref-qualifiers to assignment operators in the C++ Std

Simon Sebright – Variables with no Name

Phil Nash – C++ Extension Methods

James Grenniing – We're Different

Russel Winder – Who Needs C++ When You Have D and Go?

Anders Schau Knatten – OMG Science!