Stories from BIND9 refactoring

Dealing with code that can drink legally

Witold Kręcicki

February 3, 2019

What is old code?

What is old code?

- Software developer in mid 90's in Poland

What is old code?

- Software developer in mid 90's in Poland
- Write a simple 'phonebook' application

What is old code?

- Software developer in mid 90's in Poland
- Write a simple 'phonebook' application

56 — Area code

6552355 — Subscriber number

What is old code?

- Software developer in mid 90's in Poland
- Write a simple 'phonebook' application

56 — Area code

6552355 — Subscriber number

- That's easy!

```
const char areas[][16] = {"","","","","","","","","","","","","Krakow","".
const char* phonearea(const char* num) {
    int i, area;
    if (strlen(num) != 9) {
        return "";
    }
    for (i = 0; i < 9; i++) {
        if (num[i]<'0' && num[i]>'9') {
            return "";
        }
    }
    area = 10*(num[0]-'0') + num[1]-'0';
    return areas[area];
}
```

```
const char areas[][16] = {"","","","","","","","","","","","","Krakow",""....};
const char* phonearea(const char* num) {
    int i, area;
    if (strlen(num) != 9) {
        return "";
    }
    for (i = 0; i < 9; i++) {
        if (num[i]<'0' && num[i]>'9') {
            return "";
        }
    }
    if (!strncmp(num, "601", 3)) {
        return "M-ERA";
    }
    if (!strncmp(num, "602", 3)) {
        return "M-PLUS";
    }
    if (!strncmp(num, "501", 3)) {
        return "M-IDEA";
    }

    area = 10*(num[0]-'0') + num[1]-'0';
    return areas[area];
}
```

```
const char areas[][16] = {"","","","","","","","","","","","","Krakow",""....};
const char* phonearea(const char* num) {
    int i, area;
    if (strlen(num) != 9) {
        return "";
    }
    for (i = 0; i < 9; i++) {
        if (num[i]<'0' && num[i]>'9') {
            return "";
        }
    }
    if (!strncmp(num, "601", 3) || !strncmp(num, "603", 3)) {
        return "M-ERA";
    }
    if (!strncmp(num, "602", 3) || !strncmp(num, "604", 3)) {
        return "M-PLUS";
    }
    if (!strncmp(num, "501", 3) || !strncmp(num, "502", 3)) {
        return "M-IDEA";
    }

    area = 10*(num[0]-'0') + num[1]-'0';
    return areas[area];
}
```

```
const char areas[][16] = {"","","","","","","","","","","","","Krakow",""....};
const char* phonearea(const char* num) {
    int i, area;
    if (strlen(num) != 9) {
        return "";
    }
    for (i = 0; i < 9; i++) {
        if (num[i]<'0' && num[i]>'9') {
            return "";
        }
    }
    if (!strncmp(num, "50", 2) || !strncmp(num, "60", 2) || !strncmp(num, "69", 2) || !strncmp(num, "51", 2)) {
        int res, op;
        res = mobile_db_lookup(num, &op);
        if (res == 0) {
            switch (op) {
                case 26001:
                    return "T-ERA";
                    break;
                case 26002:
                    return "T-PLUS";
                    break
                case 26003:
                    return "T-ORANGE";
                    break;
            }
        }
    }
    if (!strncmp(num, "601") || !strncmp(num, "603")) {
        return "M-ERA";
    }
    if (!strncmp(num, "602") || !strncmp(num, "604")) {
        return "M-PLUS";
    }
    if (!strncmp(num, "501") || !strncmp(num, "502")) {
        return "M-IDEA";
    }

    area = 10*(num[0]-'0') + num[1]-'0';
    return areas[area];
}
```

What's wrong with old code?

What's wrong with old code?

- Not badly written - just filled with technical debt

What's wrong with old code?

- Not badly written - just filled with technical debt
- No testability

What's wrong with old code?

- Not badly written - just filled with technical debt
- No testability
- It works, it performs well - "if it ain't broken don't fix it"

What's wrong with old code?

- Not badly written - just filled with technical debt
- No testability
- It works, it performs well - "if it ain't broken don't fix it"
- It won't fix itself

What's wrong with old code?

- Not badly written - just filled with technical debt
- No testability
- It works, it performs well - "if it ain't broken don't fix it"
- It won't fix itself
- Hard to understand

What's wrong with old code?

- Not badly written - just filled with technical debt
- No testability
- It works, it performs well - "if it ain't broken don't fix it"
- It won't fix itself
- Hard to understand
- High barrier to entry

BIND9

BIND9

- First commit (imported from CVS):

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS
- BIND9 is older than:

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS
- BIND9 is older than:
  - The Matrix (1999)

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS
- BIND9 is older than:
  - The Matrix (1999)
  - Agile Manifesto (2001)

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS
- BIND9 is older than:
  - The Matrix (1999)
  - Agile Manifesto (2001)
  - Test Driven Development (2003)

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS
- BIND9 is older than:
  - The Matrix (1999)
  - Agile Manifesto (2001)
  - Test Driven Development (2003)
  - Linux 2.2 (1999)

BIND9

- First commit (imported from CVS):

```
commit 7ee52cc7d195433bb8f55972e2a8ab29668f7bce
Author: Bob Halley <source@isc.org>
Date:   Mon Aug 17 22:05:58 1998 +0000
```

- Replacement for BIND8 Buggy Internet Name Daemon
- Design by contract
- No exploits just possibility of DoS
- BIND9 is older than:
  - The Matrix (1999)
  - Agile Manifesto (2001)
  - Test Driven Development (2003)
  - Linux 2.2 (1999) not to mention 2.6 with NPTL (2003)

BIND is like an onion

BIND is like an onion

- Reference implementation

BIND is like an onion

- Reference implementation
- If there's an RFC - BIND has it

BIND is like an onion

- Reference implementation
- If there's an RFC - BIND has it
- Support for servers on dialup connections

BIND is like an onion

- Reference implementation
- If there's an RFC - BIND has it
- Support for servers on dialup connections
- Small team and not many external contributors (barrier to entry)

BIND is like an onion

- Reference implementation
- If there's an RFC - BIND has it
- Support for servers on dialup connections
- Small team and not many external contributors (barrier to entry)
- That's how old and complex code is created

LET'S DO SCIENCE!

LET'S DO SCIENCE!

- How to define complex code?

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity
- Number of linearly independent paths through a program's (function) source code

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity
- Number of linearly independent paths through a program's (function) source code
- The more complex code the harder it is to understand it, and the more impossible it becomes to test it

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity
- Number of linearly independent paths through a program's (function) source code
- The more complex code the harder it is to understand it, and the more impossible it becomes to test it
- Below 10 - OK

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity
- Number of linearly independent paths through a program's (function) source code
- The more complex code the harder it is to understand it, and the more impossible it becomes to test it
- Below 10 - OK
- Below 20 - Worrying

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity
- Number of linearly independent paths through a program's (function) source code
- The more complex code the harder it is to understand it, and the more impossible it becomes to test it
- Below 10 - OK
- Below 20 - Worrying
- Above 20 - Bad

LET'S DO SCIENCE!

- How to define complex code?
- McCabe cyclomatic complexity
- Number of linearly independent paths through a program's (function) source code
- The more complex code the harder it is to understand it, and the more impossible it becomes to test it
- Below 10 - OK
- Below 20 - Worrying
- Above 20 - Bad
- Above 40 - Horrible

LET'S DO APPLIED SCIENCE!

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

- 2500LOC

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

- 2500LOC
- Complexity 474

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

- 2500LOC
- Complexity 474
- Tons of gotos, going backward, going into switch statements, etc.

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

- 2500LOC
- Complexity 474
- Tons of gotos, going backward, going into switch statements, etc.
- It wasn't always that bad - started of at around 100

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

- 2500LOC
- Complexity 474
- Tons of gotos, going backward, going into switch statements, etc.
- It wasn't always that bad - started of at around 100
- Lots of mobile operators, lots of humps of the DNS camel...

LET'S DO APPLIED SCIENCE!

- pmccabe - simple tool to calculate McCabe cyclomatic complexity of functions
- BIND9 source code, bin/named/query.c

```
static isc_result_t
query_find(ns_client_t *client, dns_fetchevent_t *event, dns_rdatatype_t qtype);
```

- 2500LOC
- Complexity 474
- Tons of gotos, going backward, going into switch statements, etc.
- It wasn't always that bad - started of at around 100
- Lots of mobile operators, lots of humps of the DNS camel...
- Hold my beer!

Disclaimer

Disclaimer

- I'm not saying that my approach is perfect

Disclaimer

- I'm not saying that my approach is perfect
- I'm not saying that my approach is even good

Disclaimer

- I'm not saying that my approach is perfect
- I'm not saying that my approach is even good
- Most of things I'll state are obvious, but I really wish someone would tell me them before I started...

Disclaimer

- I'm not saying that my approach is perfect
- I'm not saying that my approach is even good
- Most of things I'll state are obvious, but I really wish someone would tell me them before I started...
- Wise people learn from mistakes

Disclaimer

- I'm not saying that my approach is perfect
- I'm not saying that my approach is even good
- Most of things I'll state are obvious, but I really wish someone would tell me them before I started...
- Wise people learn from mistakes, smart people learn from others mistakes

Disclaimer

- I'm not saying that my approach is perfect
- I'm not saying that my approach is even good
- Most of things I'll state are obvious, but I really wish someone would tell me them before I started...
- Wise people learn from mistakes, smart people learn from others mistakes
- The following is a collection of my mistakes

How to start?

How to start?

- Read the code

How to start?

- Read the code
- Read the code

How to start?

- Read the code
- Read the code
- Read the code once again

How to start?

- Read the code
- Read the code
- Read the code once again
- You probably won't understand all the possible flows - Mr. McCabe predicted that - but you'll see the 'outline'

How to start?

- Read the code
- Read the code
- Read the code once again
- You probably won't understand all the possible flows - Mr. McCabe predicted that - but you'll see the 'outline'
- Don't start unless for every piece of the code you can at least tell what it's doing

Crisis

Crisis

- It's a lost cause, let's rewrite it from scratch!

Crisis

- It's a lost cause, let's rewrite it from scratch!
- ... can you guarantee that what you'll write will perform at least as well as what you have now?

Crisis

- It's a lost cause, let's rewrite it from scratch!
- ... can you guarantee that what you'll write will perform at least as well as what you have now?
- ... can you guarantee that the behaviour won't change?

Crisis

- It's a lost cause, let's rewrite it from scratch!
- ... can you guarantee that what you'll write will perform at least as well as what you have now?
- ... can you guarantee that the behaviour won't change?
- ... how many new bugs will you introduce?

Crisis

- It's a lost cause, let's rewrite it from scratch!
- ... can you guarantee that what you'll write will perform at least as well as what you have now?
- ... can you guarantee that the behaviour won't change?
- ... how many new bugs will you introduce?
- ... do you have enough tests to verify it?

Crisis

- It's a lost cause, let's rewrite it from scratch!
- ... can you guarantee that what you'll write will perform at least as well as what you have now?
- ... can you guarantee that the behaviour won't change?
- ... how many new bugs will you introduce?
- ... do you have enough tests to verify it?
- ... do you have the budget?

Crisis

- It's a lost cause, let's rewrite it from scratch!
- ... can you guarantee that what you'll write will perform at least as well as what you have now?
- ... can you guarantee that the behaviour won't change?
- ... how many new bugs will you introduce?
- ... do you have enough tests to verify it?
- ... do you have the budget?

Making the progress

Making the progress

- Cut it into smaller pieces:

Making the progress

- Cut it into smaller pieces: take something that looks like a 'function'

Making the progress

- Cut it into smaller pieces: take something that looks like a 'function' ... and move it to a separate function

Making the progress

- Cut it into smaller pieces: take something that looks like a 'function' ... and move it to a separate function
- Optionally - create a state structure to pass between functions

```
if (event != NULL) {
        /*
         * We're returning from recursion.  Restore the query context
         * and resume.
         */
        want_restart = false;

        rpz_st = client->query.rpz_st;
        if (rpz_st != NULL &&
            (rpz_st->state & DNS_RPZ_RECURSING) != 0)
        {
                CTRACE(ISC_LOG_DEBUG(3), "resume from RPZ recursion");

                is_zone = rpz_st->q.is_zone;
                authoritative = rpz_st->q.authoritative;
                RESTORE(zone, rpz_st->q.zone);
                RESTORE(node, rpz_st->q.node);
                RESTORE(db, rpz_st->q.db);
                RESTORE(rdataset, rpz_st->q.rdataset);
                RESTORE(sigrdataset, rpz_st->q.sigrdataset);
                qtype = rpz_st->q.qtype;

                if (event->node != NULL)
                        dns_db_detachnode(event->db, &event->node);
                SAVE(rpz_st->r.db, event->db);
                rpz_st->r.r_type = event->qtype;
                SAVE(rpz_st->r.r_rdataset, event->rdataset);
                query_putrdataset(client, &event->sigrdataset);
(...)
```

```
static isc_result_t
query_resume(query_ctx_t *qctx) {
        isc_result_t result;
        dns_name_t *tname;
        isc_buffer_t b;

        qctx->want_restart = false;

        qctx->rpz_st = qctx->client->query.rpz_st;
        if (qctx->rpz_st != NULL &&
            (qctx->rpz_st->state & DNS_RPZ_RECURSING) != 0)
        {
                CCTRACE(ISC_LOG_DEBUG(3), "resume from RPZ recursion");

                qctx->is_zone = qctx->rpz_st->q.is_zone;
                qctx->authoritative = qctx->rpz_st->q.authoritative;
                RESTORE(qctx->zone, qctx->rpz_st->q.zone);
                RESTORE(qctx->node, qctx->rpz_st->q.node);
                RESTORE(qctx->db, qctx->rpz_st->q.db);
                RESTORE(qctx->rdataset, qctx->rpz_st->q.rdataset);
                RESTORE(qctx->sigrdataset, qctx->rpz_st->q.sigrdataset);
                qctx->qtype = qctx->rpz_st->q.qtype;

                if (qctx->event->node != NULL)
                        dns_db_detachnode(qctx->event->db, &qctx->event->node);
                SAVE(qctx->rpz_st->r.db, qctx->event->db);
                qctx->rpz_st->r.r_type = qctx->event->qtype;
                SAVE(qctx->rpz_st->r.r_rdataset, qctx->event->rdataset);
                query_putrdataset(qctx->client, &qctx->event->sigrdataset);
(...)
```

Stick to your job

Stick to your job

Remember that:

- Your job is not to optimize the code

Stick to your job

Remember that:

- Your job is not to optimize the code
- Your job is not to fix horrible bugs

Stick to your job

Remember that:

- Your job is not to optimize the code
- Your job is not to fix horrible bugs
- Your job is not to re-write pieces that look like they can be rewritten

Stick to your job

Remember that:

- Your job is not to optimize the code
- Your job is not to fix horrible bugs
- Your job is not to re-write pieces that look like they can be rewritten
- Your job is to refactor the code by cutting the function into smaller pieces

Stick to your job

Remember that:

- Your job is not to optimize the code
- Your job is not to fix horrible bugs
- Your job is not to re-write pieces that look like they can be rewritten
- Your job is to refactor the code by cutting the function into smaller pieces
- I know it's tempting to do fix things, but it's really not the time

Stick to your job

Remember that:

- Your job is not to optimize the code
- Your job is not to fix horrible bugs
- Your job is not to re-write pieces that look like they can be rewritten
- Your job is to refactor the code by cutting the function into smaller pieces
- I know it's tempting to do fix things, but it's really not the time
- Make comments, write bug reports, put post-it notes on your monitor but do not try to fix anything now

Don't be smart

Don't be smart

- If a code block looks vaguely similiar to another code block that you just cut out - make a second function that might have exactly the same content

Don't be smart

- If a code block looks vaguely similiar to another code block that you just cut out - make a second function that might have exactly the same content
- If a function can be easily simplified - that's fine, you'll do it later

Don't be smart

- If a code block looks vaguely similiar to another code block that you just cut out - make a second function that might have exactly the same content
- If a function can be easily simplified - that's fine, you'll do it later
- If some code is not reachable - you'll cut it out later

Don't be smart

- If a code block looks vaguely similiar to another code block that you just cut out - make a second function that might have exactly the same content
- If a function can be easily simplified - that's fine, you'll do it later
- If some code is not reachable - you'll cut it out later
- tl;dr; be a dumb code-cutting monkey

Work slowly

Work slowly

- You might miss something that's important

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason ... and you have to start from the beginning because you can't figure out what's wrong

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason ... and you have to start from the beginning because you can't figure out what's wrong
- Cut one piece

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason ... and you have to start from the beginning because you can't figure out what's wrong
- Cut one piece - commit

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason ... and you have to start from the beginning because you can't figure out what's wrong
- Cut one piece - commit - compile

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason ... and you have to start from the beginning because you can't figure out what's wrong
- Cut one piece - commit - compile - run all possible tests

Work slowly

- You might miss something that's important
- You might not notice that the piece of code you just cut out has some side effects on the global function state ...and only realize that something's wrong 3 days later when DNS64 test fails for no apparent reason ... and you have to start from the beginning because you can't figure out what's wrong
- Cut one piece - commit - compile - run all possible tests
- Don't take shortcuts or you'll pay

Write unit tests

Write unit tests

- You now have a bunch of small, simple, testable functions

Write unit tests

- You now have a bunch of small, simple, testable functions
- And you know how the code works and what to expect out of it

Write unit tests

- You now have a bunch of small, simple, testable functions
- And you know how the code works and what to expect out of it
- So drop everything and write unit tests for the functions you've just created

Write unit tests

- You now have a bunch of small, simple, testable functions
- And you know how the code works and what to expect out of it
- So drop everything and write unit tests for the functions you've just created seriously, there will never be the better time to do this

Write unit tests

- You now have a bunch of small, simple, testable functions
- And you know how the code works and what to expect out of it
- So drop everything and write unit tests for the functions you've just created seriously, there will never be the better time to do this
- Seriously, just do it - I didn't and now I really regret it

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something
- Now is the time to fix bugs you've found

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something
- Now is the time to fix bugs you've found
- Now is the time to optimize

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something
- Now is the time to fix bugs you've found
- Now is the time to optimize
- Now is the time to merge similiar functions

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something
- Now is the time to fix bugs you've found
- Now is the time to optimize
- Now is the time to merge similiar functions
- The result is probably not perfect, but it's good enough to work on further

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something
- Now is the time to fix bugs you've found
- Now is the time to optimize
- Now is the time to merge similiar functions
- The result is probably not perfect, but it's good enough to work on further
- Examples - qname minimization, modules

Taking care of the post-it notes

- Now you can fix thing not worrying that you'll break something
- Now is the time to fix bugs you've found
- Now is the time to optimize
- Now is the time to merge similiar functions
- The result is probably not perfect, but it's good enough to work on further
- Examples - qname minimization, modules
- Remember to measure your code regularly

Questions?