Designing Koha for Support-ability

Barton Chittenden

ByWater Solutions

Hi, My name is Barton Chittenden,

I do technical support for ByWater Solutions.

I'd like to talk about Koha from the
perspective of someone who does technical support,
and show how that fits in with the design of Koha.

The tech support perspective

* Tech support is two handshakes away from patrons.
* Tech support people see problems

Tech support is two handshakes
away from patrons

Tech support people see problems.
No one calls tech support to say
"My system is working perfectly".

This came to me one day when I
a couple of years after I started;
I was showing koha off to someone,
and everything I showed them worked
really well... and I realized that
this wasn't the Koha that I saw every
day. ... so I want to apologize in
advance; this talk isn't "Koha, the
good parts version.". I only get to
read the parts of the book where Miracle
Max's potion wears off, Innigo
Montoya gets gravely wounded, and
the white horses get away from Fezzik.

I also work with ByWater, which is
largely a U.S. support company;
and I don't have the perspective of
doing tech support in anything other
than English -- I suspect that there
may be some communication challenges
in other languages.

So, before we go any further,

Pressing phone buttons in a phone system?

… Pressing phone buttons in a phone system?

Keep your hands up if you experienced

A long wait on hold

... A long wait on hold

What about ...

Hearing the phrase "I'm sorry you're having trouble with your ..."

... Hearing the phrase
"I'm sorry you're having trouble with your ..."
... cable mode, router, phone service...

Ok, I'm going to give your arms a break ... put your hands down if

You were happy with your tech support call

... You were happy with your tech support call

You felt like the person on the other end of the line was really listening

... You felt like the person on the other end of the line was really listening

You *really* felt like
the person on the other end
of the call was sorry.

... you *really* felt like the person
on the other end of the call was sorry.

Ok, go ahead and put your hands down.

The underlying problem is that when you call
into an enormous phone center, you're going to
be talking to someone who is taking 150 calls a day,
and who has been turned into a robot,
dealing with the same problems over and over again.

This isn't a good experience for anyone involved.

One of the reasons for this is that...

"A computer lets you make more mistakes faster than any other invention with the possible exceptions of handguns and Tequila."

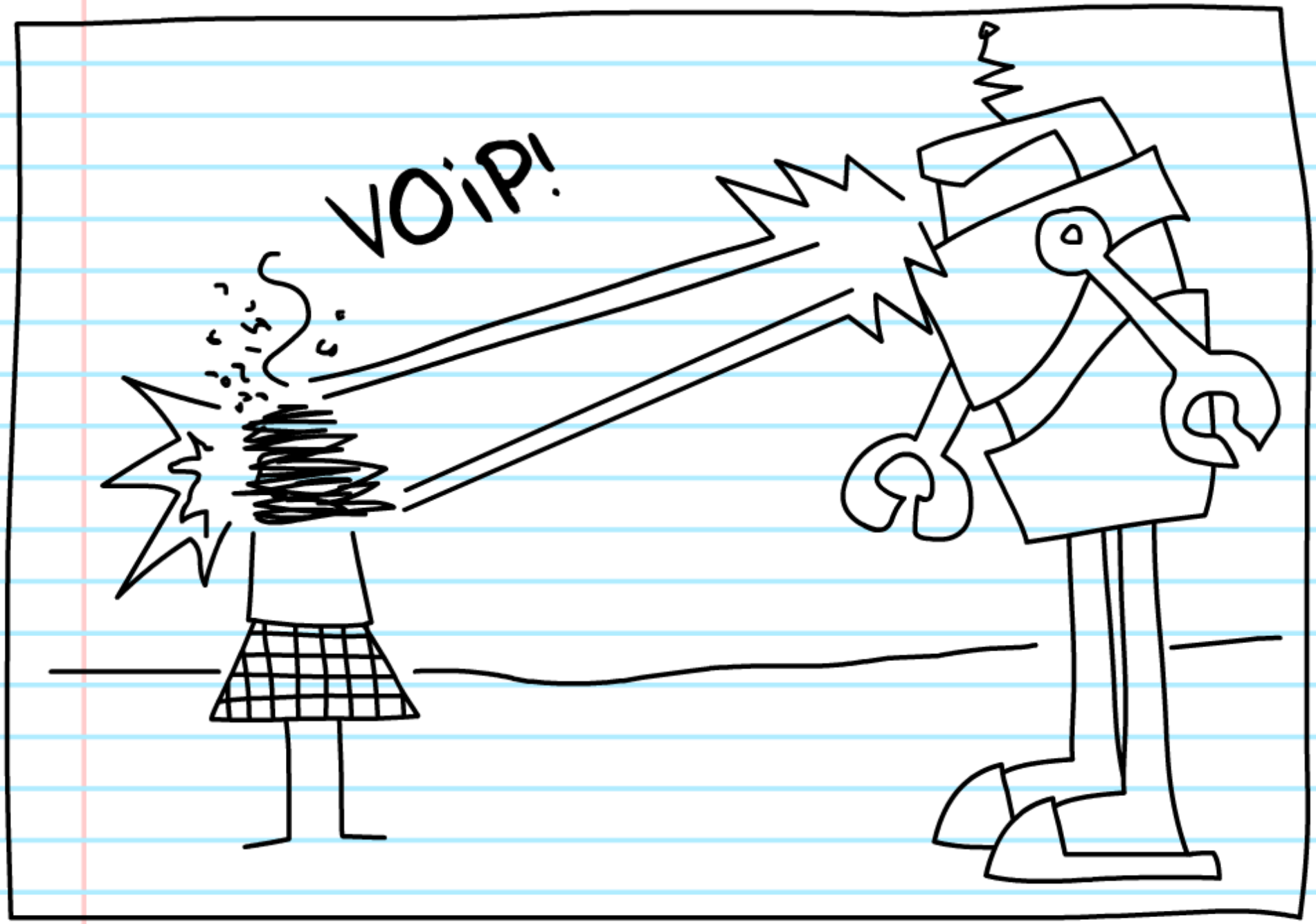
-- Mitch Ratcliffe

Computers can cause problems faster than people
can solve them.

In a previous life, I did support for a VoIP product, a lot like Vonage.
We had a feature that we rushed into production,

This feature didn't get adequately tested at scale,

This feature didn't get adequately tested at scale,
and it ended up crashing the entire phone network


VOiP!

every 20 seconds.

It took operations 3 days to figure this out.

It took operations 3 days to figure this out.

The whole time, I was left saying "I'm sorry you're having trouble with your VoIP."

The whole time, I was left saying
"I'm sorry you're having trouble with your VoIP.
Yes, we are aware of this issue;
I don't have a resolution at this time".

Those were the three longest days of my life in tech support.

Those were the three longest days of my life in tech support.

It sucked my compassion dry.

It sucked my compassion dry.

^ If VOiP melt down is here

If the taking down an entire telecom VoIP system every 20 seconds is on the extreme Left,

Koha lives about here ^

Koha lives about here ^

Then koha is about two thirds of the way to the right.

Because Koha is a mature open source project with an excellent QA team, and because it's deployed it's deployed on lots of of small servers rather than one large server, it's unlikely that we will get hit with a catastrophic affecting everyone at once.

However, one or more bad bugs in a major release can create a slow motion version of a support melt-down, where support staff is not overwhelmed in a single day, but worn down over a period of time.

Tech support is a human endeavor, and it only works when the people doing it have the time, energy and patience to pay attention.

The effects on our compassion, our ability to listen and communicate effectively are diminished.

We want to get here ^

This talk is aimed at looking at some of the design decisions that would make Koha a little easier to support, either for a systems librarian at a site which has their own Koha instance, or for a support vendor. In essence, our goal is to be able to support koha with kindness, compassion and to be entirely present to those who we're helping.

The Elements of technical support

* Communication

What do support people do?

* Communication

  At ByWater, we use the term 'reference interview'
  -- similar to the usage at a reference desk.

  We focus on broad, non-leading questions, trying
  to determine what the partner wants and expects,
  and how that differs from their experience.

  Nick used the example yesterday of a library
  patron talking to a reference librarian, asking
  about clouds, but who really wanted to know about
  abstract art.

  ... so a big part of our job is getting to
  the partner's wants and needs.

The Elements of technical support

* Communication
* Classifying issues

Classifying issues:

We try to

* Assess the urgency and importance of the issue

* Categorize the issue to find similar problems
  -- perhaps the problem has already been solved.

* Figure out if this is a bug or a feature

* Figure out what additional questions to ask

* Find the right person to solve the problem

The Elements of technical support

* Communication
* Classifying issues
* Replicating issues

Replicating issues

If we can see the problem and make it happen,
we know that we have effective communication
This also makes troubleshooting easier

The Elements of technical support

* Communication
* Classifying issues
* Replicating issues
* Trouble-shooting

Trouble-shooting

We want to get to the root cause of the issue,
and find a solution, or at least a work-around
in the case of a bug.

What are the challenges?

Let's look at some of the challenges that make each of these elements difficult.

Communication

* Confusing interfaces

An example of this would be the rules for lost statuses.
The behavior of lost items depends on

* System preferences

* Authorized values

* Arguments in the cron file

* Lost status of the item itself

The rules that define when a lost item fee is charged are
not *explicitly* defined anywhere.

Communication

* Confusing interfaces
* Interfaces that are difficult to describe

The pages under Acquisitions are visually very similar.
It's not clear how these pages relate to each other,
and it's often difficult to tell how a partner got
to a certain screen.

Communication

* Confusing interfaces
* Interfaces that are difficult to describe
* Inconsistent interfaces

Koha divides circulation notices into "advanced notices" and "overdue notices".

Overdue notices have one syntax for showing item data, and advance notices have another.

Using the syntax for overdue notices rather than advance notices is a *very* common error.

Classifying problems

It's not always easy to see from the initial description of a problems.

Bread crumbs, Perl file names, and labels should all agree

Perl file names and error messages are not indexed or search-able in bugzilla

There can be a dis-connect between the structure of Koha and the structure of Bugzilla

Hard to figure out of a problem is a bug or misunderstanding of a feature

Good error messages make categorization easier, but these are hidden in plack. When a software error happens under plack, all you see is a small message on the screen that says "Software error"

Software error:

```
Can't call method "biblio" on an undefined value at /usr/share/koha/lib/C4/CourseReserves.pm line 880, <DATA> line 751.
```

For help, please send mail to the webmaster (staff@bywatersolutions.com), giving this error message and the time and date of the error.

This is what software errors looked like before we went to plack.

Now, this page looks really ugly, but there are some really important parts of this page: first, you see the actual error, this gives support staff and developers an idea of what went wrong and where.

The second thing is that there's an email link. This means that users will actually let us know when errors occur.

Replicating issues

Not obvious what configuration triggers a problem.

This makes is it easy to end up with a problem to re-create because of a hidden configuration -- or worse, that *others* can't re-create.

Don't always have initial state -- for instance holds. Capturing a hold, or running the holds queue builder change the state of the data. By the time that the problem presents itself, we're in the dark.

Intermittent issues (this is actually special case of the above)

Hard to know what the smallest amount of data is needed to replicate an issue.

Trouble-shooting

Trouble-shooting

* Scary basements

Some parts of the Koha code base, for example
the holds queue builder or the internals of
Search.pm are what Kyle calls 'Scary Basements'.

The internals are not well documented or understood
by the community at large. Sometimes these were
written by people who are no longer in the community,
In other cases the complexity has grown over time
to the point where no one knows exactly
what the functionality is.

Trouble-shooting

* Scary basements
* Intermittent problems

Sometimes we run into intermittent problems, and we don't have a large enough sample of data to spot a trend -- we essentially have to say "that this is a glitch", and move on.

Trouble-shooting

* Scary basements
* Intermittent problems
* Data problems

Some software errors are caused by data that violates Koha's data integrity -- this is increasingly common as we move toward Koha objects -- places where the previous code would have silently ignored data problems now cause fatal errors.

Trouble-shooting

* Scary basements
* Intermittent problems
* Data problems
* Reading source code

When trouble shooting gets down to the level where people in support have to read source code, this becomes very slow -- when a developer takes three hours to read a piece of code that they're not familiar with, that's probably not critical. When someone in a tech support position does the same, it's because someone is waiting to know why Koha worked this way or that, and three hours may be critical.

What can we do to make this easier?

Communication

Communication

* Make effects explicit

The user interface should be explicit about the effects of users actions.

Features should be clearly and strictly documented so that it's clear not only what the software does, but what it does *not* do.

Communication

* Make effects explicit
* Group like configurations

The configuration of like things should be grouped together, to avoid spooky action at a distance

Communication

* Make effects explicit
* Group like configurations
* Controlled vocabulary

User interfaces should use
a controlled vocabulary that
is consistent from database
table and field names, bread
crumb names, link text and
Perl controller names. If we
have a controlled vocabulary,
it becomes easier for support
to ask for clarification
using that vocabulary.

Communication

* Make effects explicit
* Group like configurations
* Controlled vocabulary
* Consistent interfaces

Consistent user interfaces
avoid support issues

Classification

Classification

* Classification should follow what the user sees

Classification

* Classification should follow what the user sees
* Error message should be sent to the screen

Wherever possible, the classification of bugs and errors should follow what the user sees.
I'm going to do a quick digression...

I had a European history teacher in high school. We were talking about the succession of kings, and the importance of male heirs...

Now I should mention that this teacher was Native American, from the Choctaw tribe.

she looked up at us with a twinkle in her eye, and said "Among the Choctaw, we don't trace lineage through the father. You don't always know who the father is. But you *always* know who the mother is.

... likewise, it may not always be clear what causes a bug,

Classification

* Classification should follow what the user sees
* Error message should be sent to the screen
* Error messages should show who had the error.

When I showed the error screen before, I mentioned the importance of the mailto link.

... the problem is that we didn't know which library the error was coming from.

Replication

Replication

* Make configuration explicit

Make configuration explicit

Replication

* Make configuration explicit
* Pull sample data easily

Make it easy to pull together data and configuration which causes the problem

Replication

* Make configuration explicit
* Pull sample data easily
* Show initial state of data

Make it easy (or at least possible)
to find the initial state of of data
before a problem occurs.

Replication

* Make configuration explicit
* Pull sample data easily
* Show initial state of data
* Show steps leading to a questions

Have the system log the steps that lead up to a problem -- essentially emitting a problem description or test plan.

Improving trouble shooting

Eliminate scary basements -- Document code, re-factor code for clarity. Shout out to Jonathan, who has moved so much code into the Koha namespace.

I can read the source code, which tells me *what* is happening, but reading the code will not tell me *why*. Document this.

Whenever code makes assumptions about data, write a query that we can use as a data checker.

Intermittent issues can be found by logging. We need to start using Koha::Logger.

Conclusion

Technical support lives with
the constant threat of computers
making more mistakes than they
can handle.

Our job is to help librarians,
and thereby to help library patrons,
and we do so by doing our best
to fully understand and field
every problem that comes our
way.

Designing for support-ability
helps librarians communicate
with their tech support people,
helps support folks to classify
and replicate issues, and
trouble-shoot.

Post script

I've been working on a support plugin at https://github.com/bywatersolutions/koha-plugin-support

Questions?