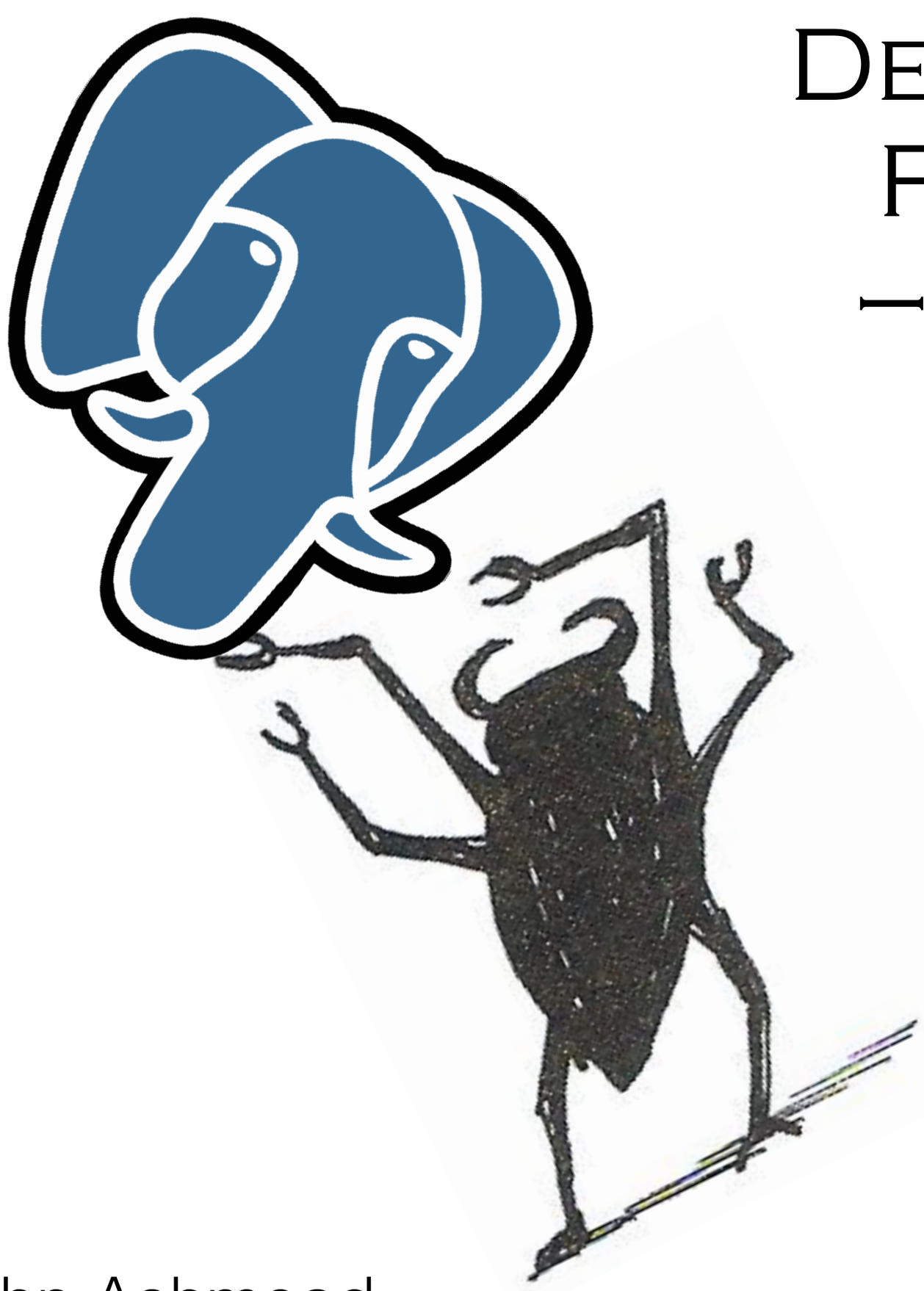
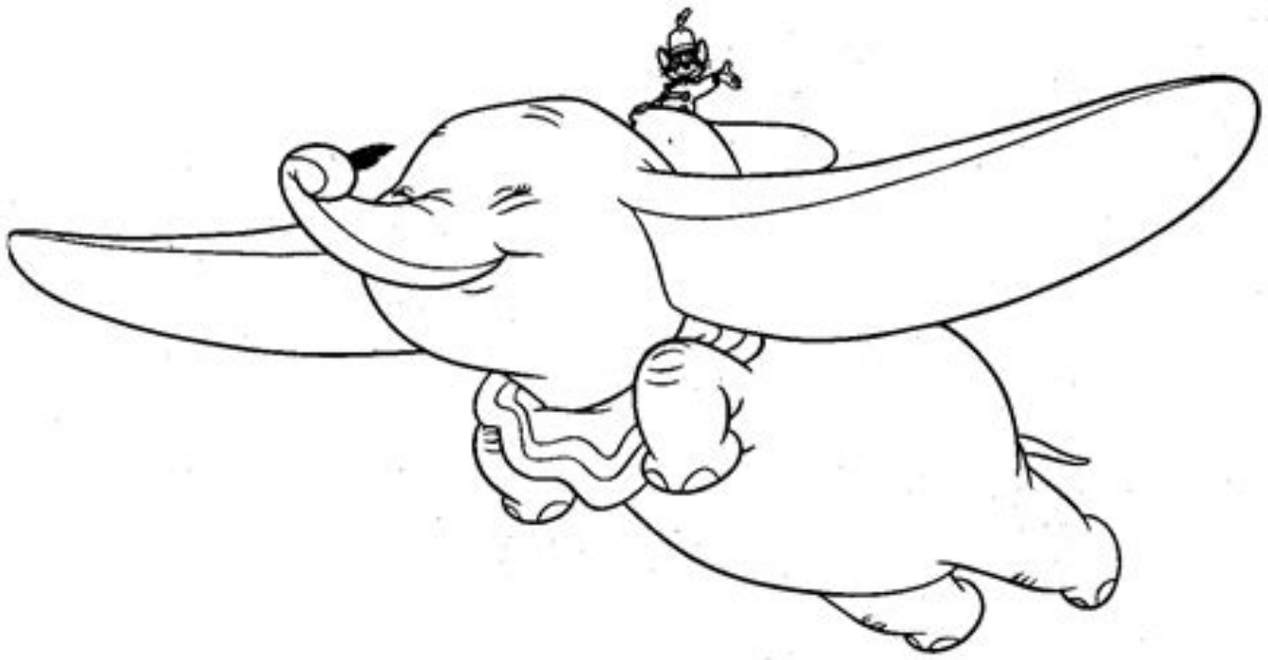


Debugging With PostgreSQL – A Strategic Approach

- Debugging is the elephant in the room
- Can take 50 to 90% of the time
- But gets 2% of the attention!
- PostgreSQL has great tools
- But they are most effective when deployed as part of an overall strategy

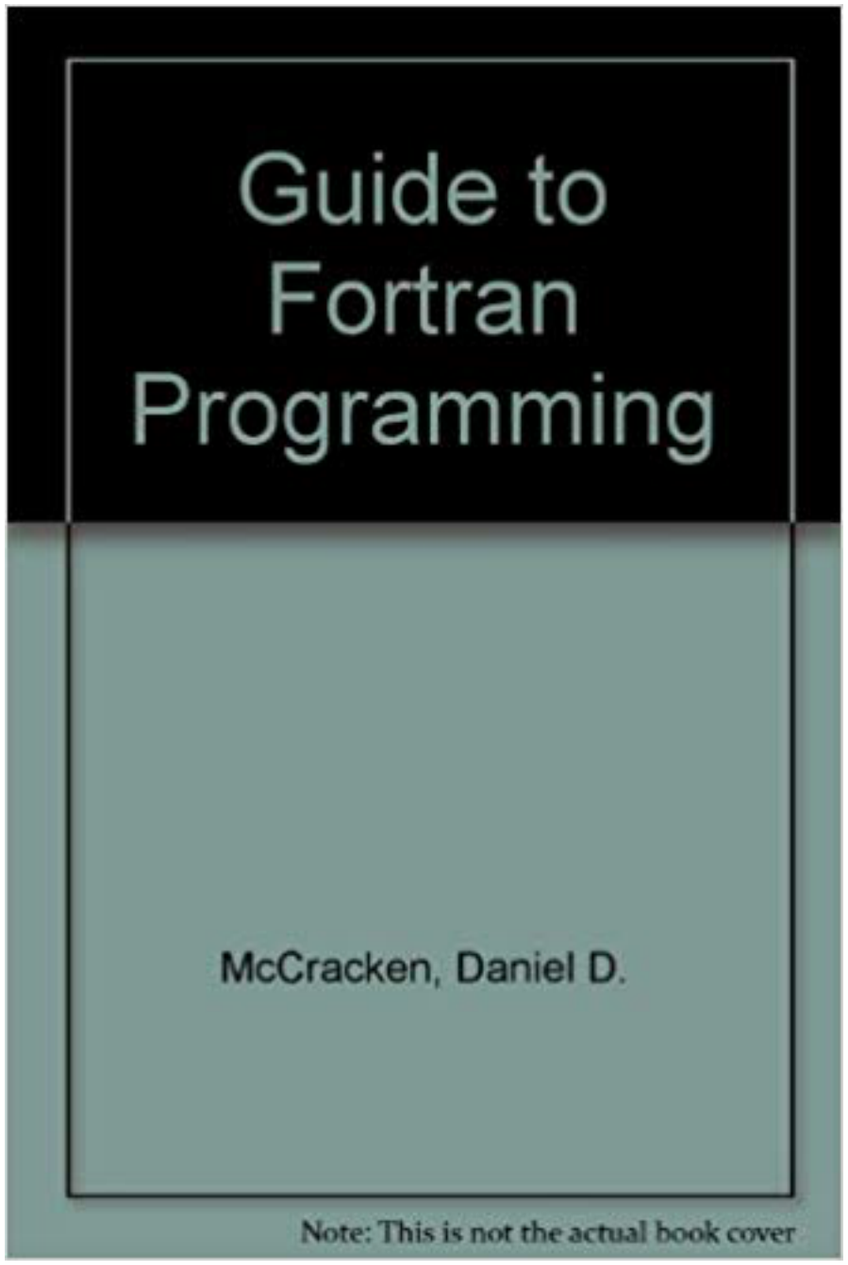
John Ashmead
john.ashmead@ashmeadsoftware.com

Cleaning up after the elephant



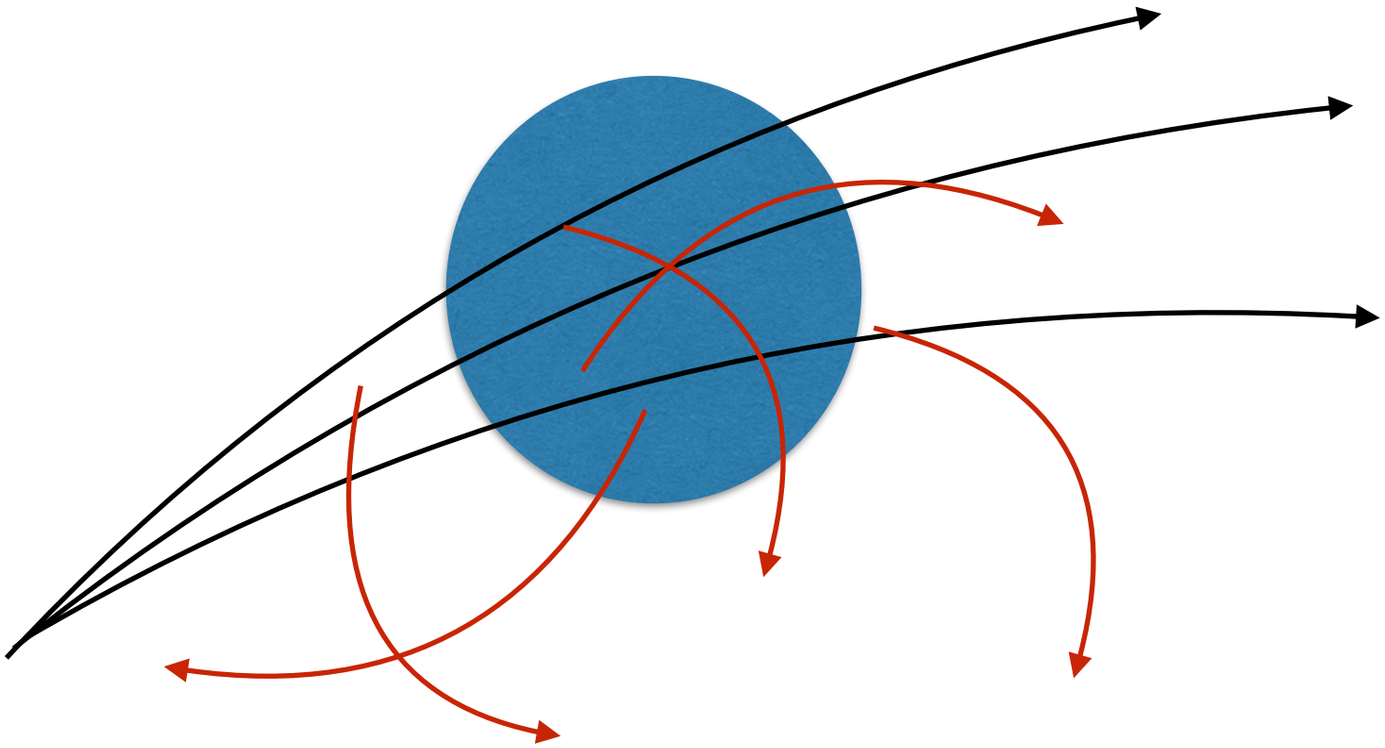
- Small shops: one or two people
- Legacy code, no documentation or handoff
- Limited resources
- Considerable time pressure

See if you can destroy this



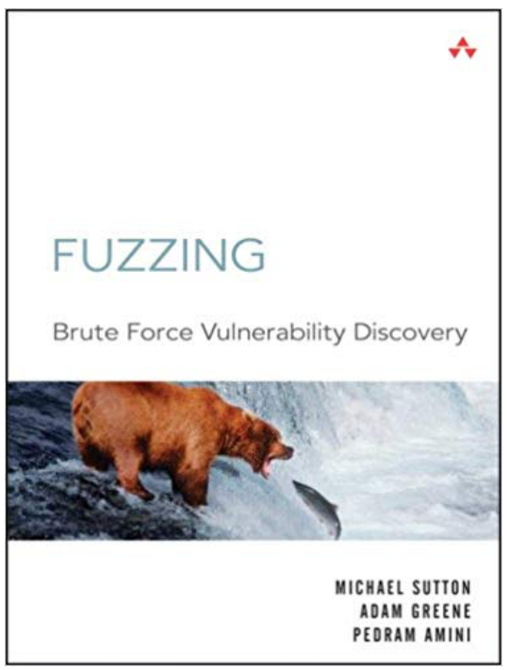
- Write from top to bottom
- Use meaningful variable names
- Throw the first one out

ROGUE PARTICLES



- Random inputs
- Generate_series()
- Fuzzing

```
SELECT * FROM generate_series(2,4);
 generate_series
-----------------
               2
               3
               4
(3 rows)
```

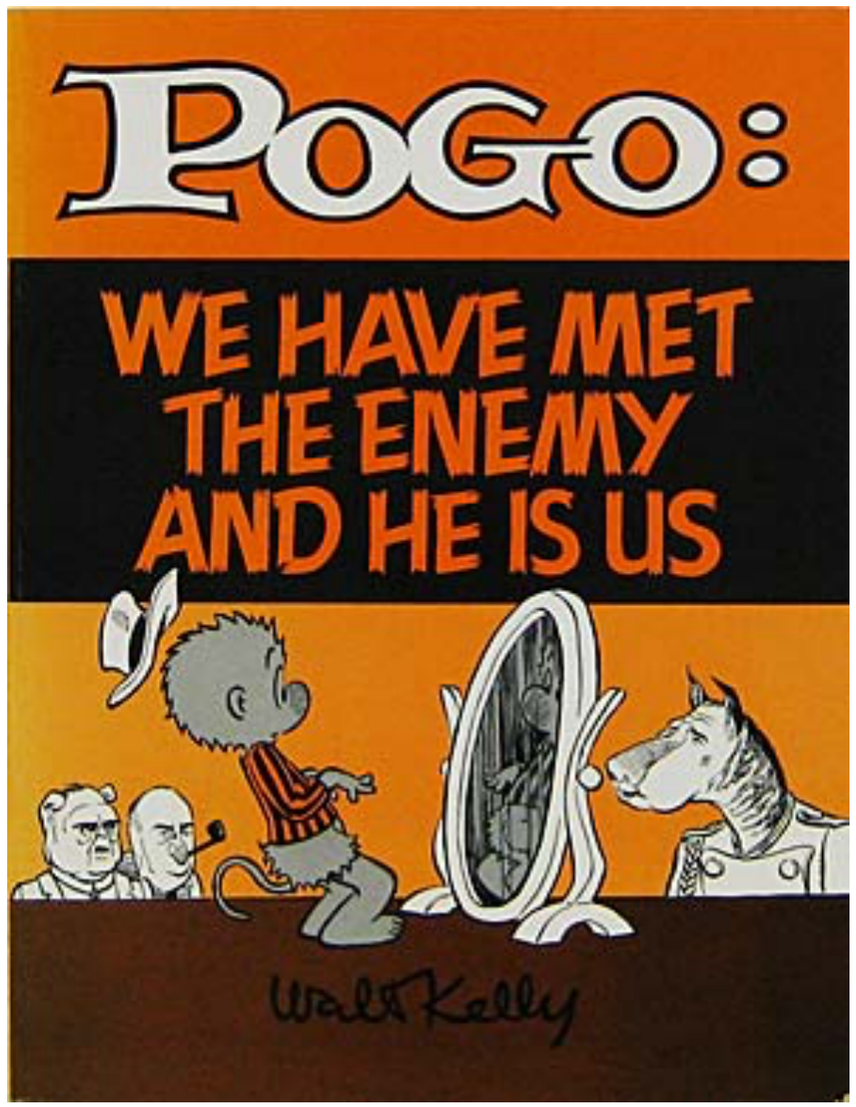


- Chaos Monkey

Strategic Debugging / Ashmead

http://timeandquantummechanics.com

The root cause



- Generally self-inflicted
- Can also be the result of bitrot,
- New requirements,
- Changing expectations, …

FOREIGN KEYS

```
CREATE TABLE cities (
        city     varchar(80) primary key,
        location point
);

CREATE TABLE weather (
        city     varchar(80) references
cities(city),
        temp_lo  int,
        temp_hi  int,
        prcp     real,
        date     date
);
```

- Correctness
- Speed
- Documentation
- Meta-programming

There are (at least) three bugs here;
you have two slides to find them!

TIMESTAMPS NOT UPDATED

```
CREATE FUNCTION timestamp_trg()
RETURNS trigger
    LANGUAGE plpgsql
    AS $$
  begin
    /* assume we have updated_at
     and created_at fields on the
     table */
    if new.created_at is null
    then
      new.created_at = now();
    end if;
    new.updated_at = now();
    return new;
  end;
$$;
```

- Add fields regardless
- Ruby-on-rails will update for you
- But more reliable in PostgreSQL
- Using simple naming convention lets you write function once

KRUNGTHEPMAHANAKHON AMONRATTANAKOSIN
MAHINTHARAYUTTHAYA MAHADILOKPHOP
NOPPHARATRATCHATHANIBURIROM
UDOMRATCHANIWETMAHASATHAN
AMONPHIMANAWATANSATHIT
SAKKATHATTIYAWITSANUKAMPRASIT

```
create domain city_t as text not null;
create domain location_t as point;
create domain fahrenheit_t as int check(value between -460 and 10000);
create domain precipitation_t as real check(value >= 0);
```

```
CREATE TABLE cities (
        city     city_t primary key,
        location location_t
);

CREATE TABLE weather (
        city       city_t references cities(city),
        temp_lo    fahrenheit_t,
        temp_hi    fahrenheit_t,
        prcp       precipitation_t,
        weather_date       date
);
```

Split Brain Problem

```
create table
patients_whose_doctors_pays (
    patient_id it_t,
    name name_t,
    address addr_t,
    doctor_id id_t references
doctors(id),
    medical_stuff
medical_stuff_t,
    ...
);
```

```
create table
patients_who_insurers_pays (
    patient_id it_t,
    name name_t,
    address addr_t,
    primary_insurance_id id_t
references insurers(id),
    medical_stuff medical_stuff_t,
    ...
);
```

```
create view patients as
    select 'doctor' as pay_type,
           patient_id, name, address, medical_stuff, ...
        from patients_whose_doctor_pays
    union all
    select 'insurance',
           patient_id, name, address, medical_stuff, ...
        from patients_whose_insurance_pays
;
```

Write perfect code

- At a conference, I asked Bjarne:
- How do I debug C++?
- "Write perfect code!"
- Annoying …
- But not without a grain of truth

Code has TWO readers

J.5. Style Guide

J.5.1. Reference Pages

J.5.1. Reference Pages

Reference pages should follow a standard layout. This allows users to find the desired information more quickly, and it also encourages writers to document all relevant aspects of a command. Consistency is not only desired among PostgreSQL reference pages, but also with reference pages provided by the operating system and other packages. Hence the following guidelines have been developed. They are for the most part consistent with similar guidelines established by various operating systems.

Name

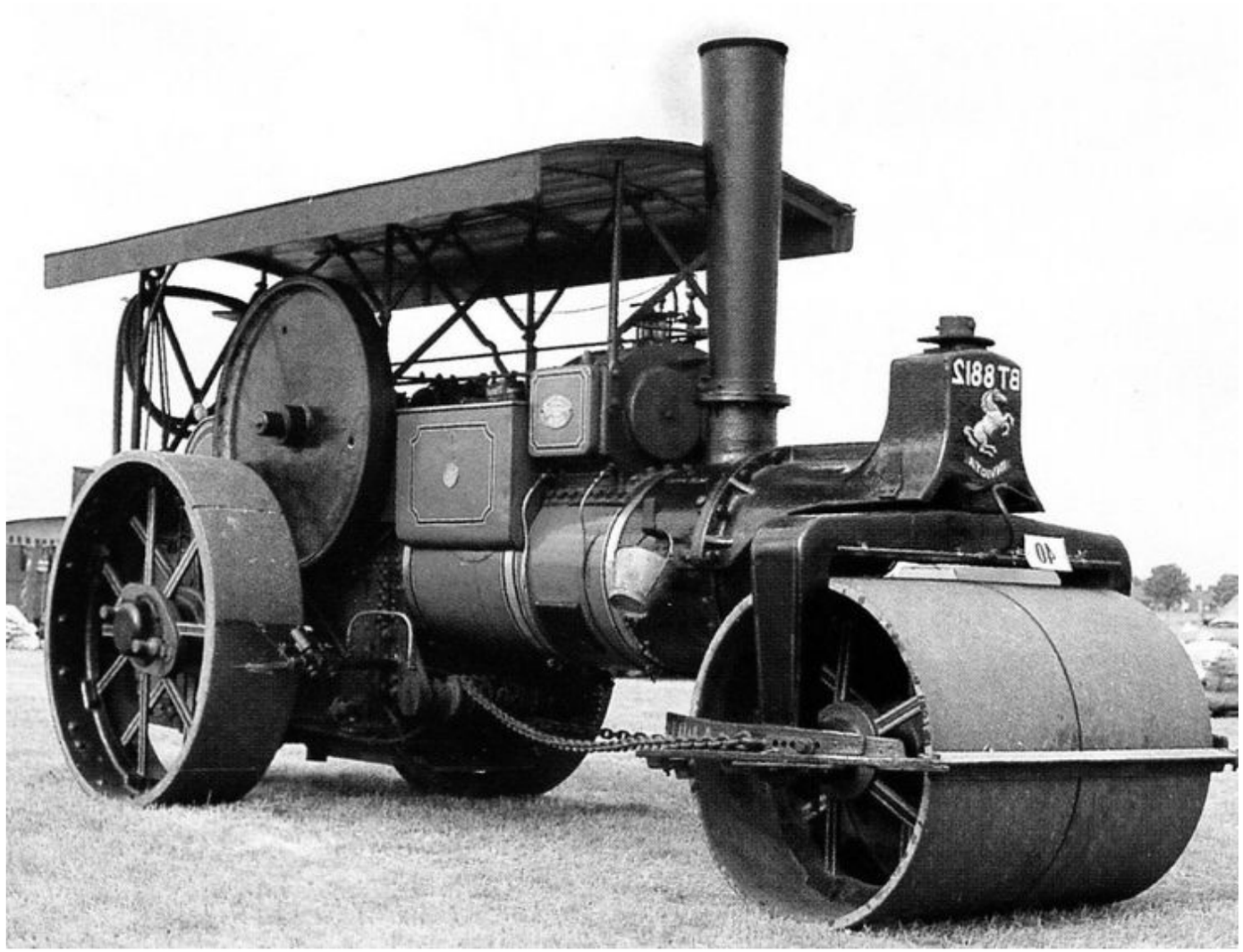
This section is generated automatically. It contains the command name and a half-sentence summary of its functionality.

- Should be clear to intended reader
- Use meaningful names
- Flow from top to bottom
- Use logical, consistent indenting
- Use appropriate comments
- Use house style

Cat in a Box

- Claw in every direction until you see daylight
- Sometimes necessary
- Fuzzing & the Chaos Monkey
- Get some sleep
- Get a 2nd pair of eyes on the problem

Linear Debugging



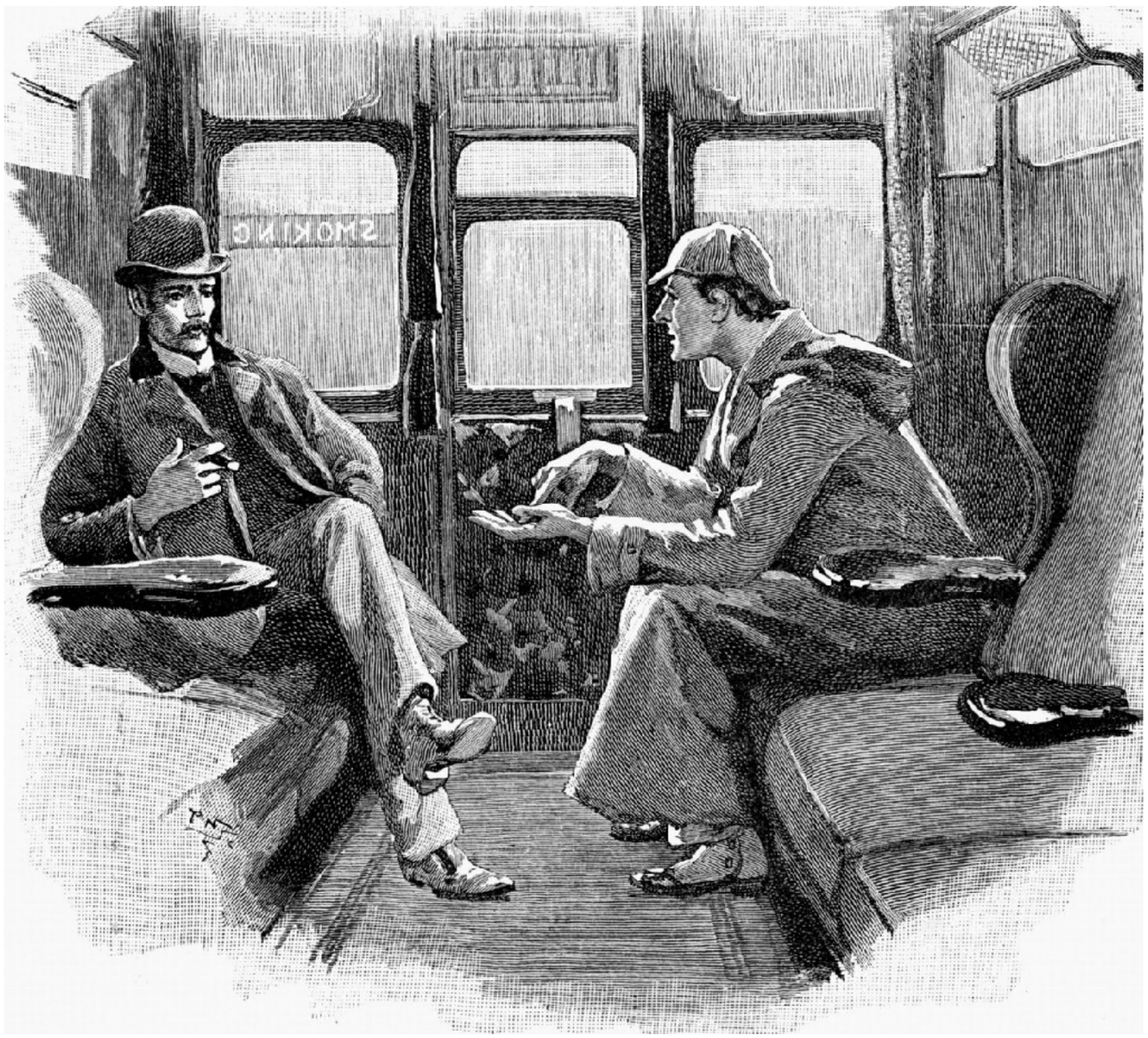
- Raise notice, print statements
- Debuggers
- Linear process, so slow
- Should usually be the **last** resort

Instrumentation

```
drop table if exists results_step1;

create temp table results_step1 as
select field1, field2, field3 from
fairly_complex_query;

get diagnostics rows_found1 = row_count;

raise notice '%: % step one rows found',
    ((extract(epoch from clock_timestamp())
        - extract(epoch from
transaction_timestamp)) * 1000)::integer,
        rows_found1;
```

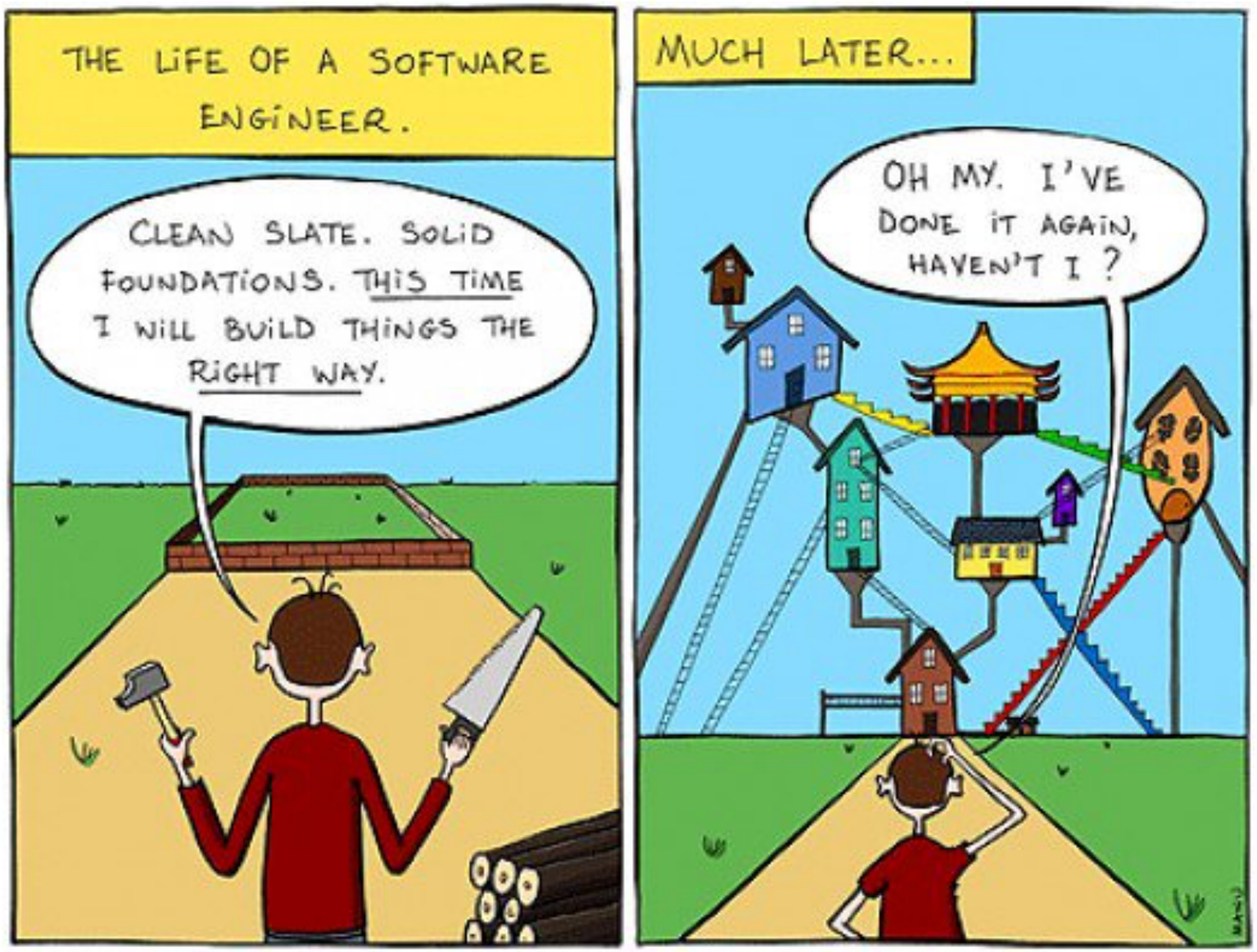
- Debuggers not all that helpful — too slow
- Preplanned & targeted has worked better for me
- As has use of unlogged tables to track intermediate steps in long computations

How to fix



- **Don't fix** right away
- Write error report/test; verify **red**
- Write fix; verify **green**
- Bugs travel in packs: look for similar problems
- Find root cause, not just immediate trigger
- See if you can prevent entirely

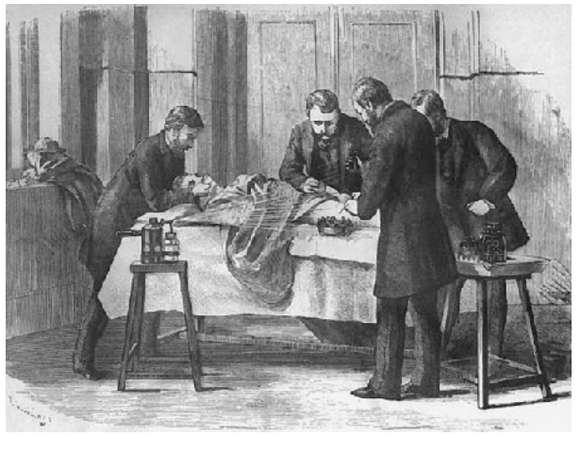
Technical Debt

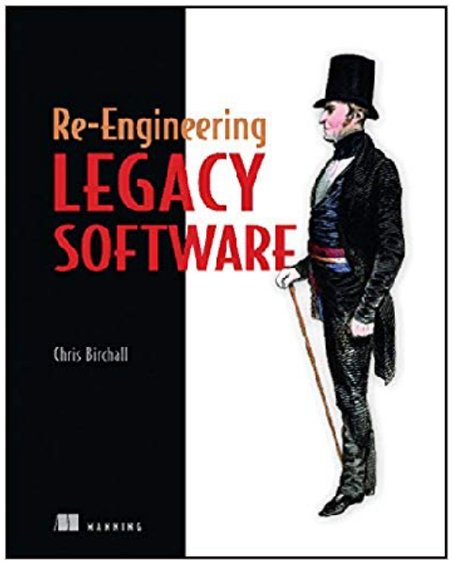


- Hard to read
- *Code smells*: lots of cut & paste, irregular indenting, patches of dead code, …
- Small changes tend to large, unwelcome side effects

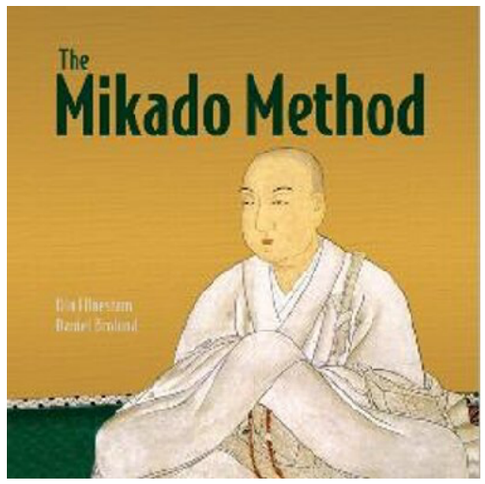
http://timeandquantummechanics.com

Repair, refactor, Rebuild





- Don't kill the patient
- Birchall - Re-Engineering Legacy Software
- Ellnestam & Brolund - Mikado Method



Systems Test

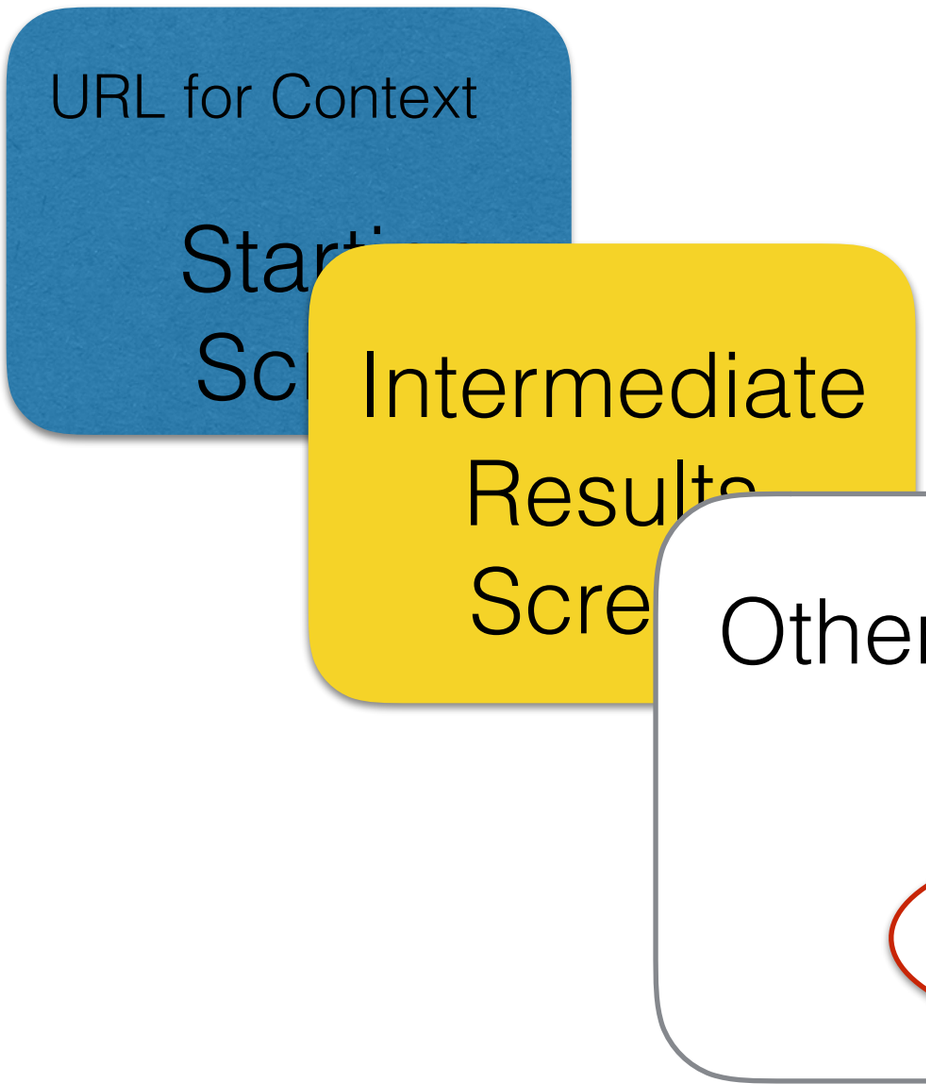
Ariane 5 Flight 501

June 4, 1996

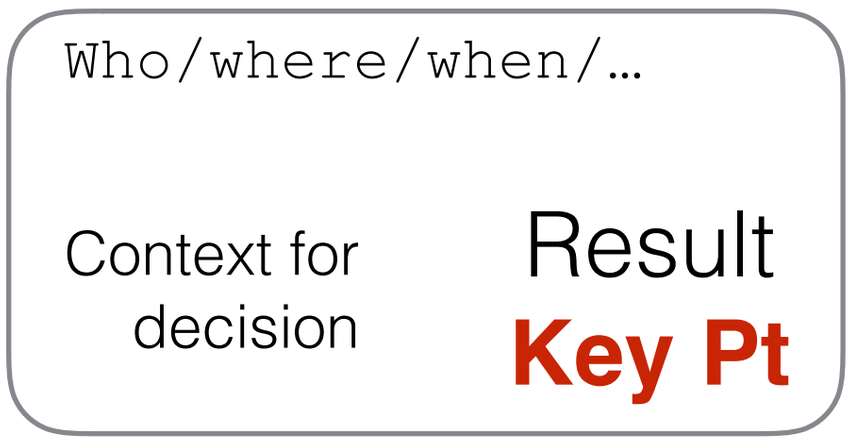
- French rocket reuses code from Ariane 4
- 5's faster engines triggers bug in arithmetic routine in flight computer
- Convert 64-bit FP to 16-bit signed int

http://sunnyday.mit.edu/accidents/Ariane5accidentreport.html

User Error – Delete User



- Educate users
- Show users what is going on
- Show them the code




```
if pat_age > age_cutoff and test_res >
max_norm then
    recommend(patient_id, followup);
else
    routine_followup(patient_id, test_id);
end if;
```

Don't Repeat Yourself

```
#!/bin/bash
# sx -- sql execute -- useful tool
for f in $*
do
  bn=`basename $f .sql`
  sql="${bn}.sql"
  out="${bn}.out"
  if [ ! -s $sql ]
  then
    echo "No $sql file"
    exit 1
  fi
  time psql -v ON_ERROR_STOP=1 -X -f $sql
> $out 2>&1
  if [ $? -ne 0 ]
  then
    echo "$sql failed:  see $out"
    exit 1
  fi
  ls -l $out
done
```

- Make
- Small shell scripts
- Function calls
- Third normal form
- Continuous automation

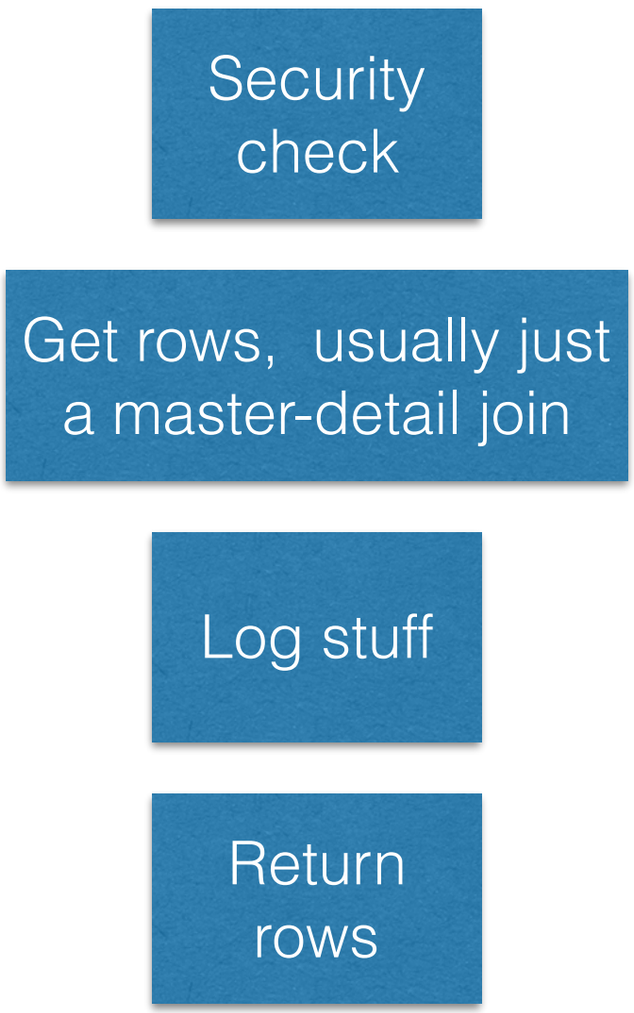
Automating common tasks

```
create table report_types (
    id primary key,
    report_name,
    title name_t,
    description desc_t,
    parameter_list jsonb, -- name + allowed values,
    fail_count count_t,
    times_run count_t,
    total_time_taken millisecond_t,
    administrators_only boolean,
    created_at tz_t,
    last_run_at tz_t,
    …
);
```

```
create table reports (
    id primary key,
    report_type_id references
report_types(id),
    parameters_used jsonb,
    run_by references users(id),
    success ok_t,
    time_taken millisecond_t,
    rows_found count_t,
    run_at tz_t
);
```

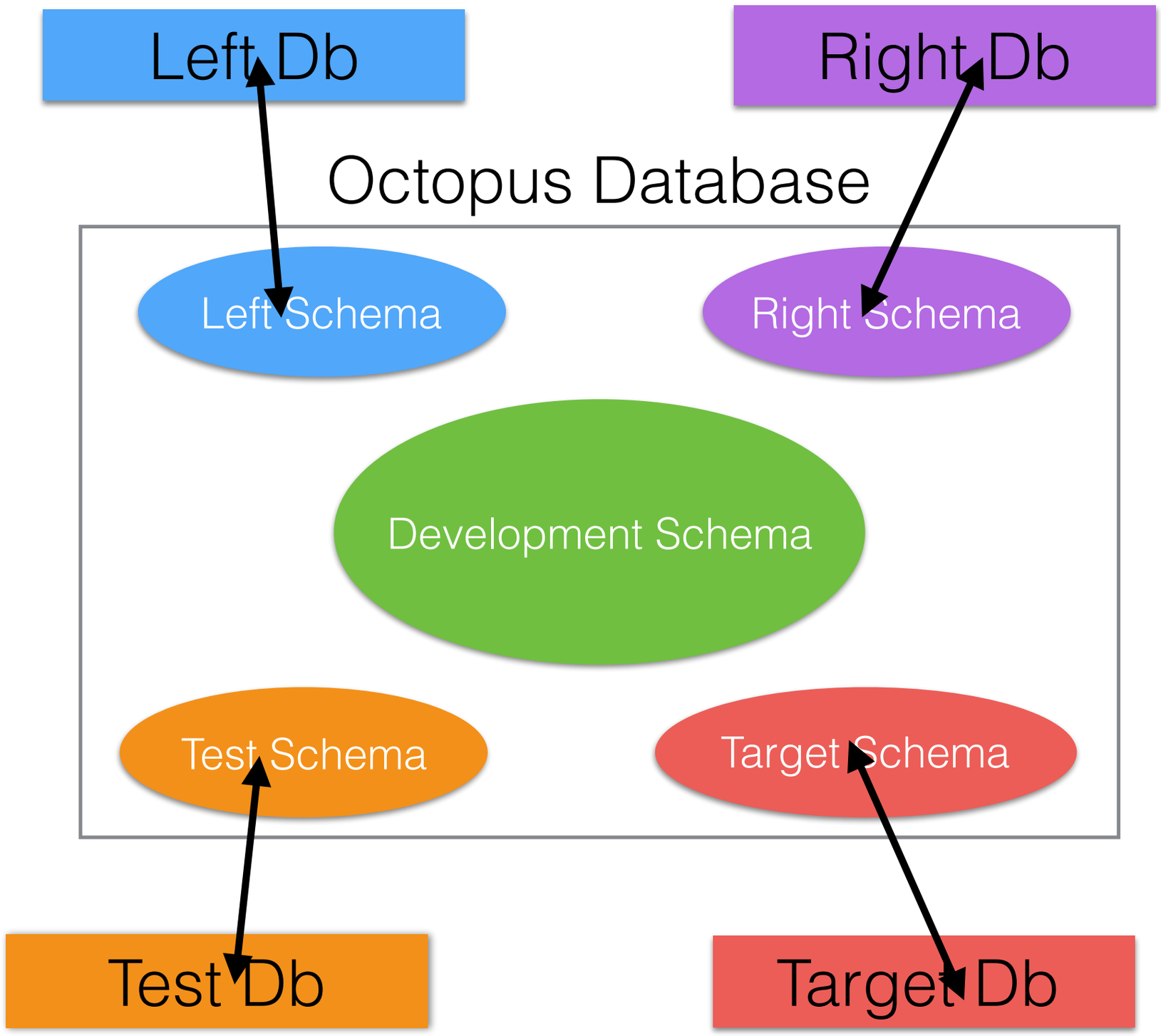
- Factor out common parts of frequent tasks
- Reports, warnings, errors,…
- Let you get an overview

Code Generators



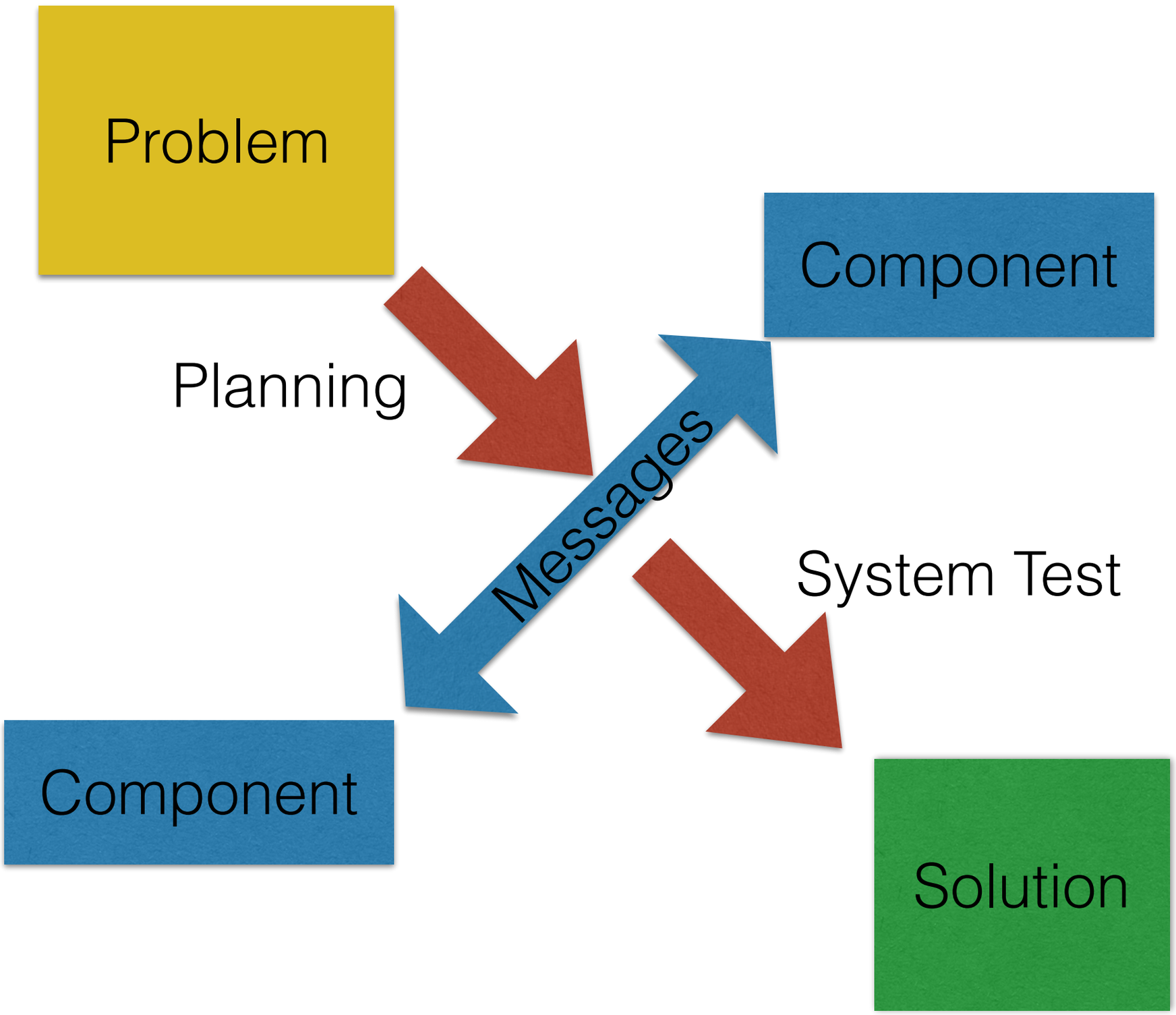
- Called in for portfolio mgmt problem
- High security reqs
- Common patterns to most queries, usually joining two tables
- Could use meta data to build SPs
- In 80 minutes the CG ground out 110,000 lines of code
- A few stragglers had to be hacked a bit or even built from scratch
- 3 month contract done in 2 weeks

Foreign Data Wrappers



- PostgreSQL good central exchange
- Pull data in, clean up, make it available
- Rapid compare of old & new versions

Divide and Conquer



- Break into parts
- Plan dev path
- Use simple message protocols
- Program defensively
- Write self-debugging code
- Debug scientifically

PLAN DEVELOPMENT PATH

As when driving,
think globally,
act locally.

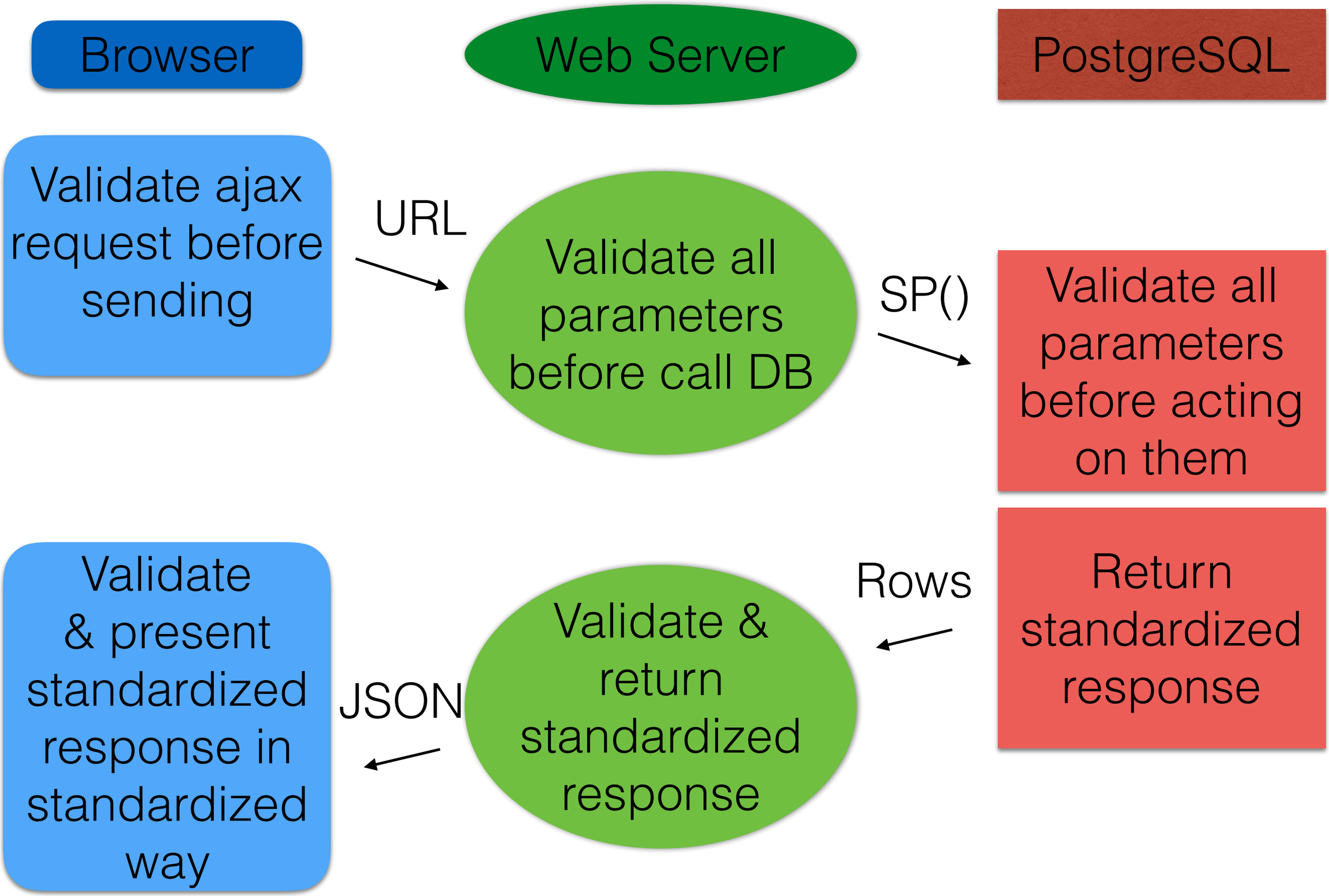
- Get something useful up quickly
- Stub later bits out
- Share with users
- Develop incrementally
- Agile has the right idea

KISS the Messenger

```
-- push out from source
-- one of the few cases (in working code)
where 'select *' is considered helpful
psql -h source_database <<!
\copy (select * from parent1) to '/tmp/
parent1.csv' csv header
\copy (select * from child1) to '/tmp/
child1.csv' csv header
…
!
-- make sure the CSVs look good, then:
-- pull in on target
psql -h target_database <<!
\copy parent1 from '/tmp/parent1.csv' csv
header
\copy child1 from '/tmp/child1.csv' csv
header
…
!
```

- Messaging key to Divide & Conquer
- KISS applies to messaging as well
- CSV is just fine for many applications
- And well supported by PostgreSQL

Program defensively



Self-debugging Code

```
create or replace function ssn_set( person_id0 people.id%type, ssn0 people.ssn%type,
debug_flag0 boolean default false)
returns ok_t as $$
-- Debugging with PostgreSQL example -- John Ashmead -- for FOSSCON 8/17/2019
declare
    person_id1 people.id%type; -- more specific than id_t
    ssn1 people.ssn%type;      -- could use ssn_t
begin
    if debug_flag0 then
        raise notice 'ssn_set(%, %)', person_id0, ssn0;
    end if;

    select id into person_id1 from people where id = person_id0;
    -- could also check number of rows found using "get diagnostics"
    if person_id1, ssn1 is null then
        raise exception 'ssn_set:  person_id0 % is not in people table', person_id0;
    end if;

    -- could trap non-unique exception here
    select id into person_id1 from people
        where ssn = ssn0 and id != person_id0;
    if person_id1 is not null then
        raise exception 'ssn_set:  ssn % is already in use by id %', ssn0, person_id0;
    end if;

    update people set ssn = ssn0 where id = person_id0;

    if debug_flag0 then
        raise notice 'ssn_set: person %: ssn changed from % to %', person_id0, ssn0, ssn1;
    end if;

    return true;
end; $$ language plpgsql;
```

Turn on tracing at will

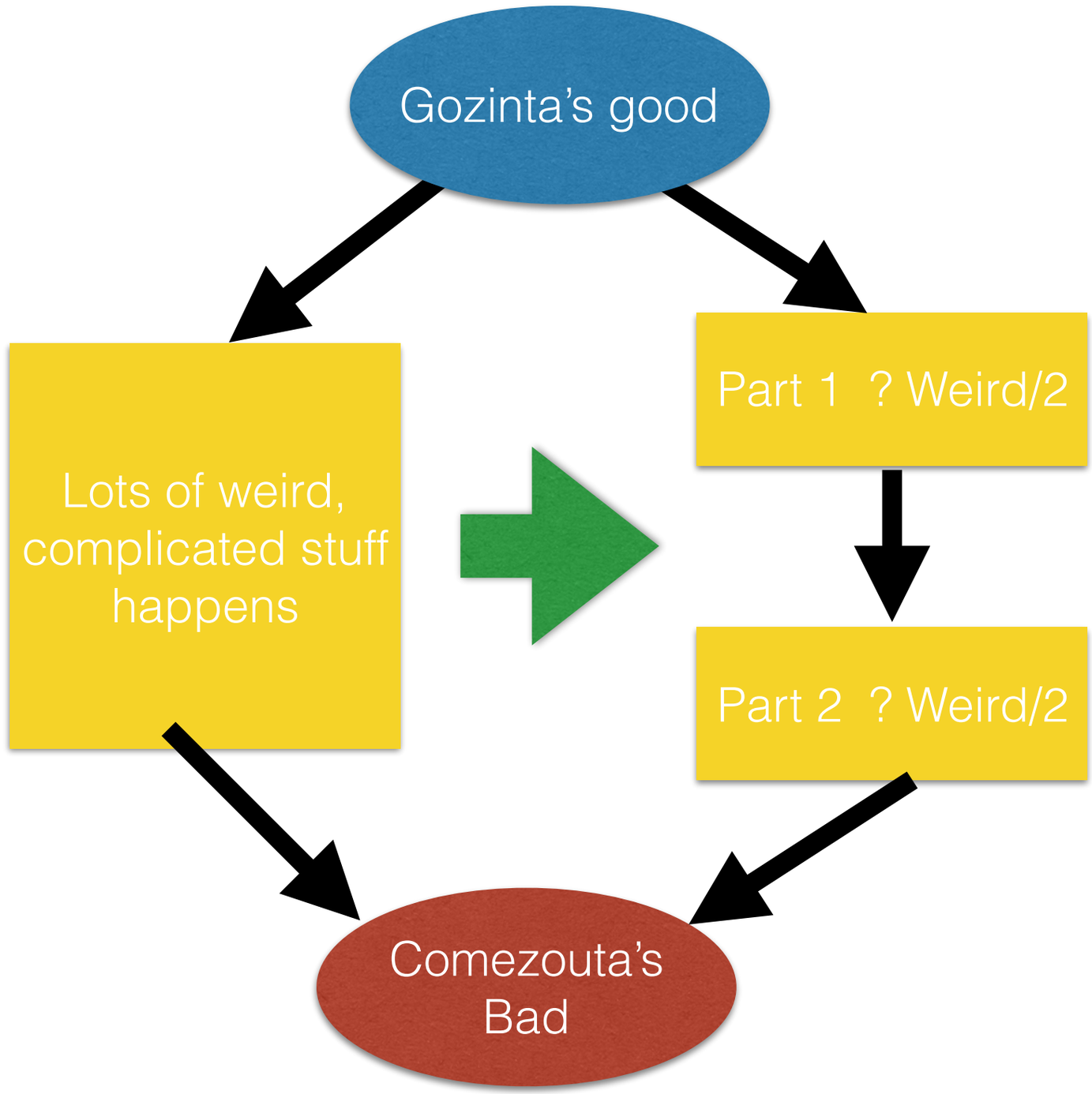
Self-documenting

and self-checking

Especially valuable in crunch mode

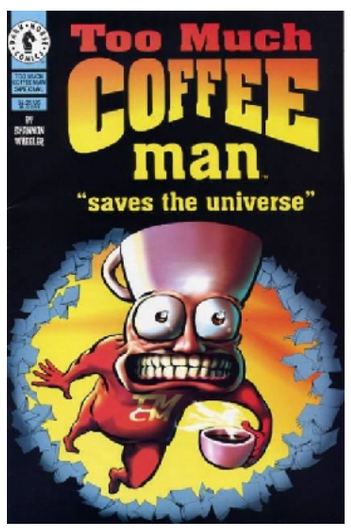
http://timeandquantummechanics.com

Gozinta's good, Comezouta's Bad



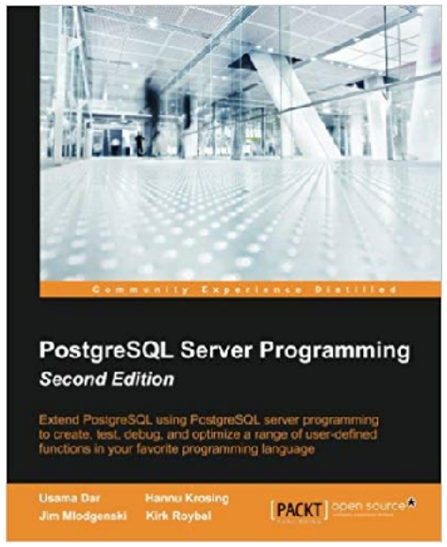
- Double check the Gozinta's anyway
- Take notes
- Have a hypothesis (or three)
- But don't guess, instrument
- Prefer **binary** search to linear; keep cutting the hiding space for bugs in half

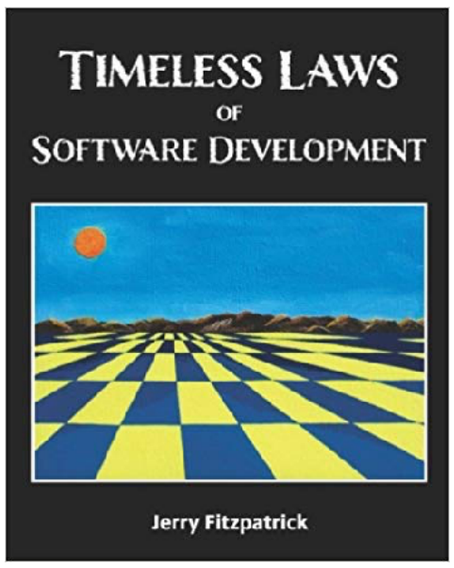
Plus/Minus

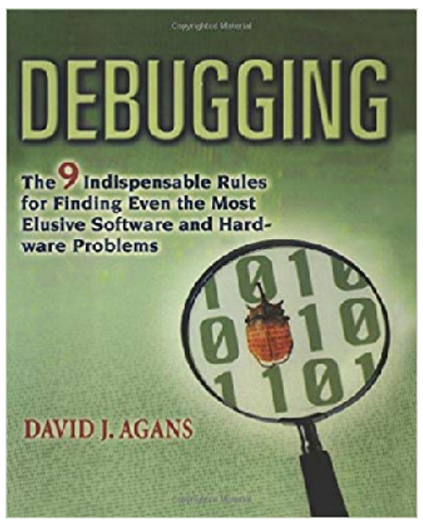


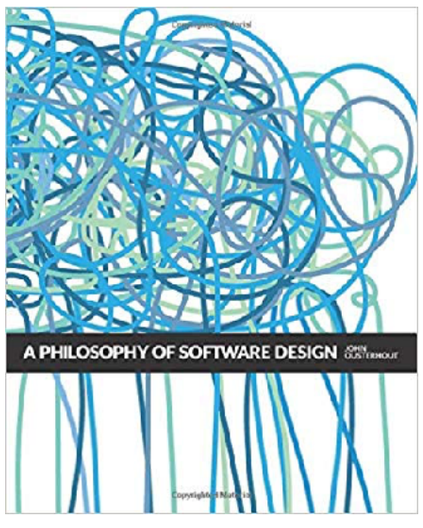
- More work upfront
- No silver bullet
- More reliability
- Less total time coding
- More interesting products

References









- St. Jerome - RTFM
- Mlodgenski - PostgreSQL Server Programming
- Fitzpatrick - Timeless Laws
- Agans - Debugging: 9 Indispensable Rules
- Ousterhout - Philosophy of Software Design