

Building a Better Team

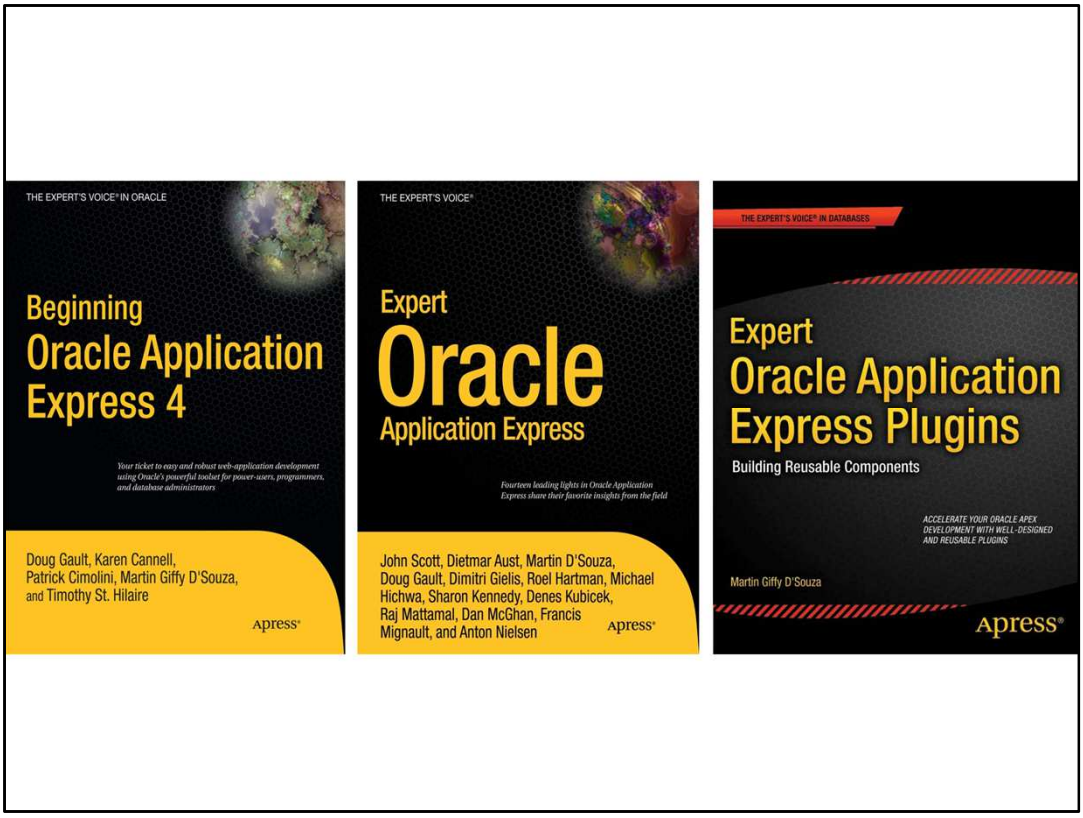
As your team grows and evolves so do some of your development problems. Most development teams struggle as they grow and try to implement top-down processes that don't always work well with developers. This presentation will guide you through proven steps and techniques to bring an unstructured development environment into a cohesive unit. The processes discussed will help you and your team develop better code, reduce development costs, and build better applications. The techniques discussed in this presentation have been successfully implemented at various organizations and have helped both APEX and PL/SQL teams.

Martin Giffy D'Souza



ORACLE ACE Director

- CTO and Co-founder at ClariFit: http://www.ClariFit.com
- Author of Oracle APEX blog: http://www.TalkApex.com
- Director for Oracle Development Tools User Group (ODTUG): http://www.odtug.com
- Oracle ACE Director



- Beginning Oracle Application Express 4: http://goo.gl/NxHoF
- Expert Oracle Application Express: http://goo.gl/tXm3P
  - More info about this book: http://jes.blogs.shellprompt.net/2011/03/30/expert-oracle-application-express
- Expert Oracle Application Express Plugins: http://goo.gl/089zi

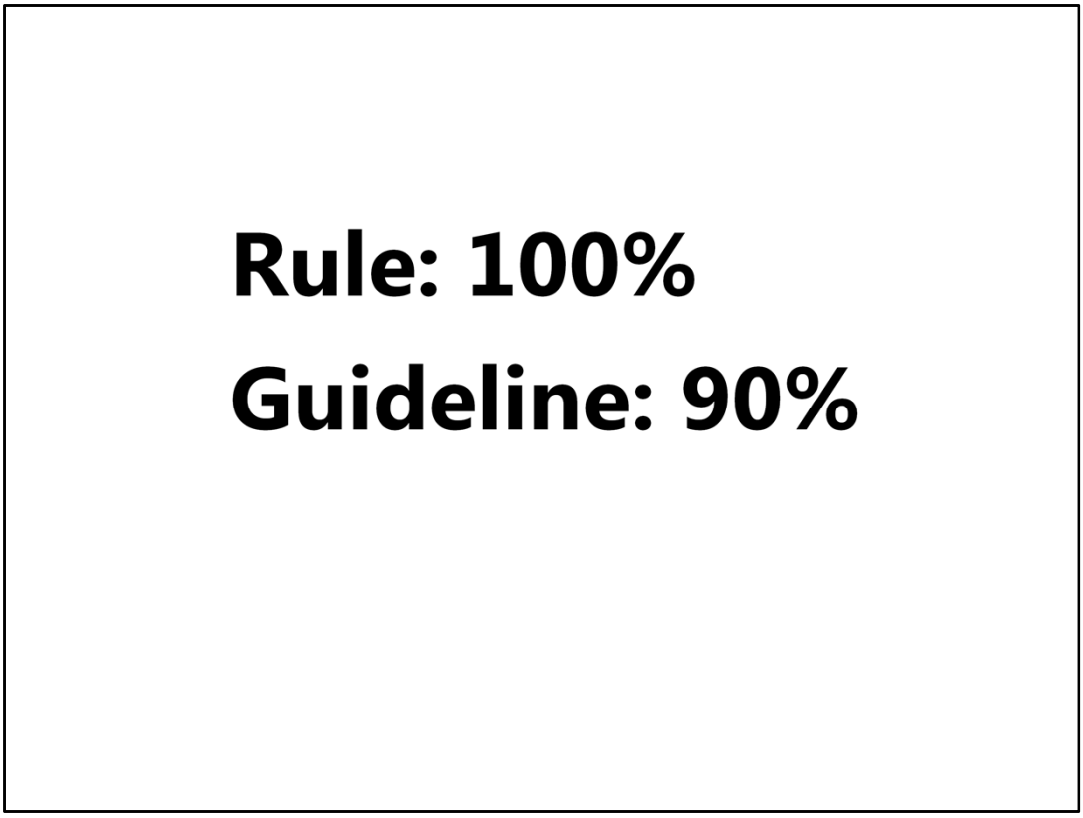


•ClariFit is a Oracle APEX and PL/SQL consulting and training firm based out of Calgary, Alberta, Canada
•For more information please visit http://www.ClariFit.com or email: info@ClariFit.com

~~Coding Standards~~ Rules & Guidelines

Coding Standards usually have the following issues:

- Written a long time ago by someone that may or may not still work in organization
    - Because of this no one understands <u>why</u> certain standards are in place
- Dry read that is usually read once and never referenced again
- Not updated



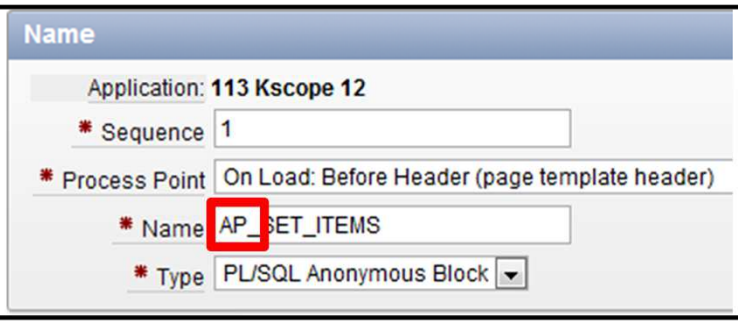
Rules & Guidelines

- Rule: Follow 100% of the time
- Guideline: Follow 90% time. In general you follow this but there are some cases when you don't follow it.
- Examples in following slides

Rule / Guideline *	Active voice sentence
Why *	Why the rule is in place
Result	Diagram/Explanation of desired outcome
How	How the result is achieved
Notes	State the non-obvious
See Also	Related R&G or links

Rules & Guidelines: Template

- All R&G are created with this template
  - First two fields are required
- Make sure R&G is an Active voice sentence
  - Removes any ambiguity
  - Ex: (incorrect) Don't use us upper case values for key words
    - So are *Select*, *SeLeCt*, … valid options?
  - Ex: (correct): Key words are lowercase
    - This leaves no opening for a different interpretation
- Keep the R/G and Why to one or two lines
- R&G final document consists of many of these R&G grouped together

Rule	Prefix application processes with “AP_”
Why	Naming consistency
Result	

Example of how to include a result from a GUI programing interface such as APEX.

Guideline	Raise exceptions in exception block
Why	Push error all the way through the application

Result	…
	`exception`
	`  when others then`
	`  …`
	`  **raise;**`
	…

For code samples try to highlight the code area by bolding it. By doing this it allows readers to know exactly what the R&G is discussing.

Implementation

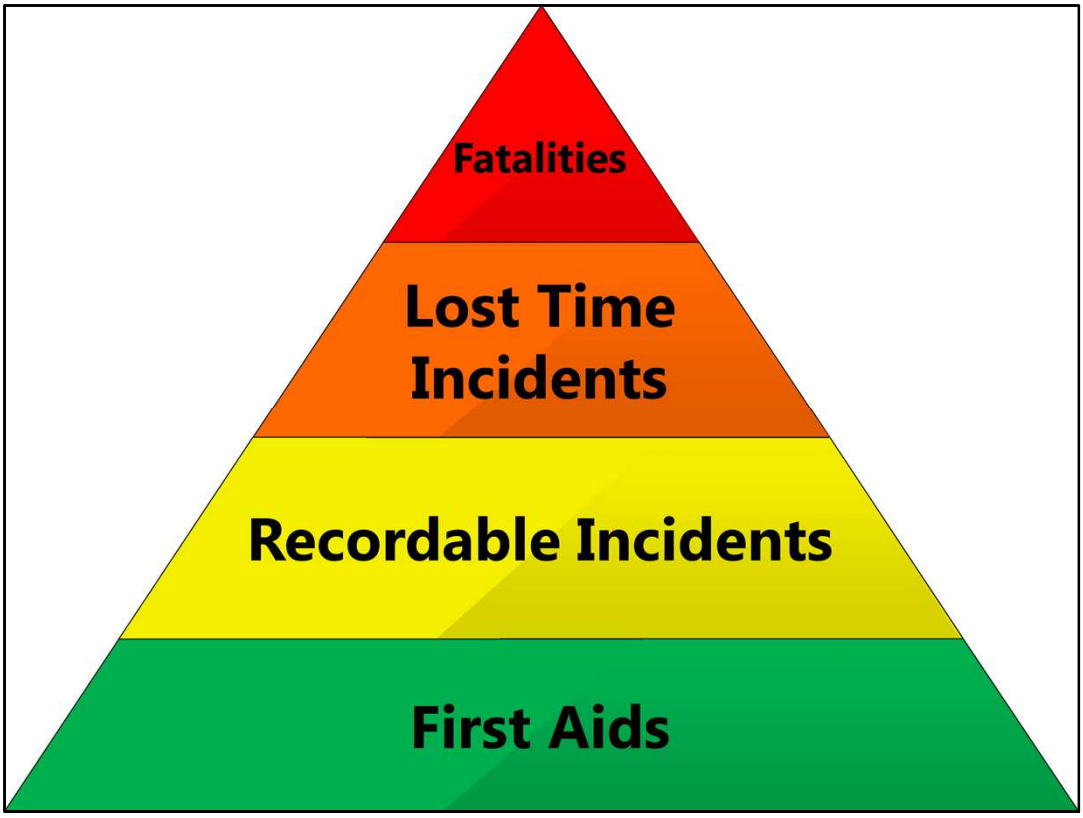
Implementation is an easy thing to do. The following “agenda” is a good one to start with and modify accordingly:

- Schedule 1 hour meetings with your team for a month two
    - During the first meeting explain what R&G are etc.
        - Come prepared with a set based on your current coding “standards”
    - Have the team vet all of them. You may find new cases come up etc.
    - By having the team agree to the R&G you have buy in since they all participated
    - If there’s a tie you, as document master, need to make final decision
- Have team consistently send you new R&G they want to see included in the document which you include for each meeting’s discussion
- Eventually you won’t need to meet once a week. Change to once a month.
- Finally all you’ll need to do is meet once a quarter

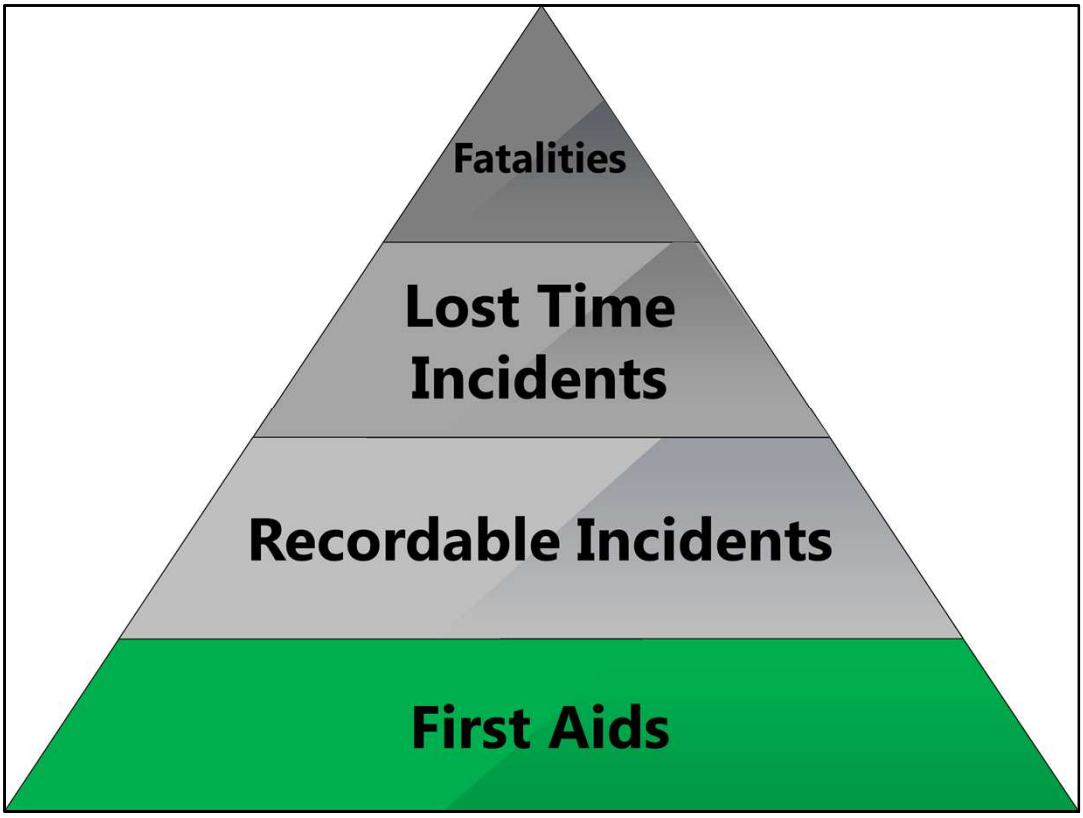


- Code reviews are not tough to do. The biggest challenge is cultural and political
    - Historically developers hate having someone else look at their code
        - Give example of civil engineers and bridge building. All bridge designs get thoroughly reviewed before they are built. If they were built with the same methodology developers (i.e. no one sees design) then they'd all fall apart
    - Managers question the time
        - 2 things happen with code reviews
        - Cross training
            - Business functionality
            - Technical skill set
        - Reduction in small bugs which will save time/money/quality etc (more on this in next sets of slides)

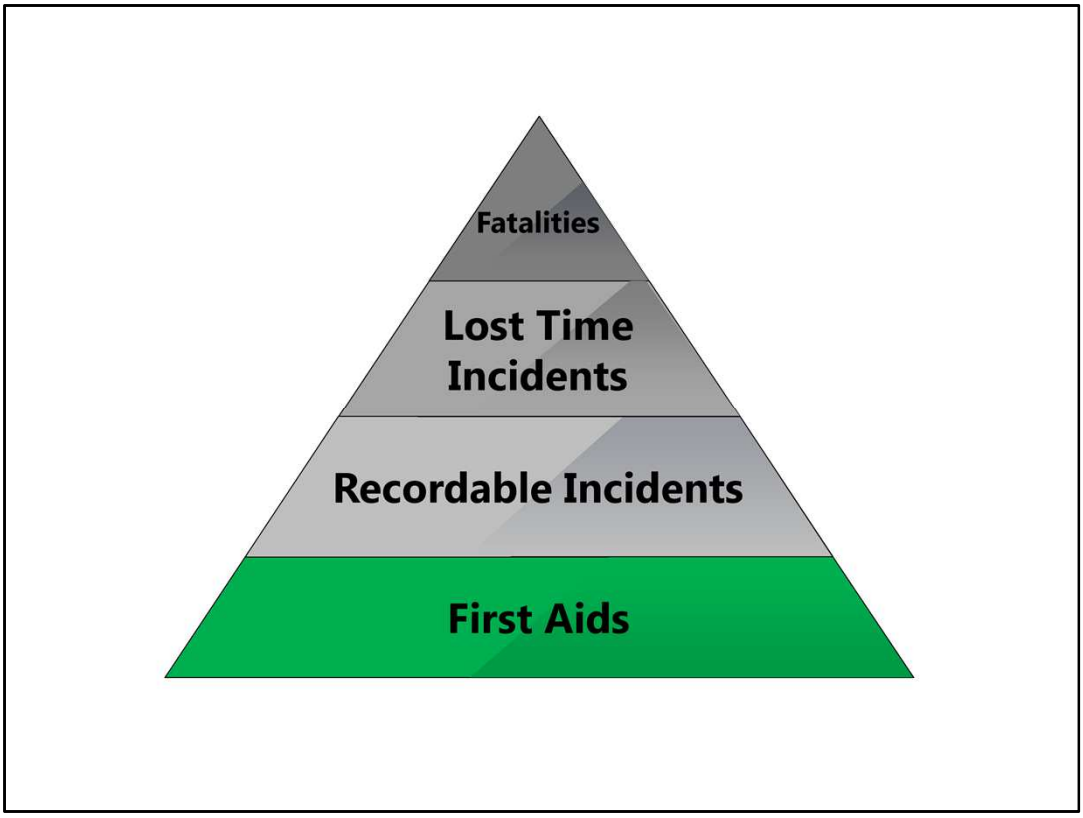
Historically
- Developers hate them
- Management doesn't like them
- No one has fun
- Talk about the advantages
    - Cross Training
    - Learn from other developers
    - Consistency



Safety triangle used in large organizations to help reduce injuries and fatalities



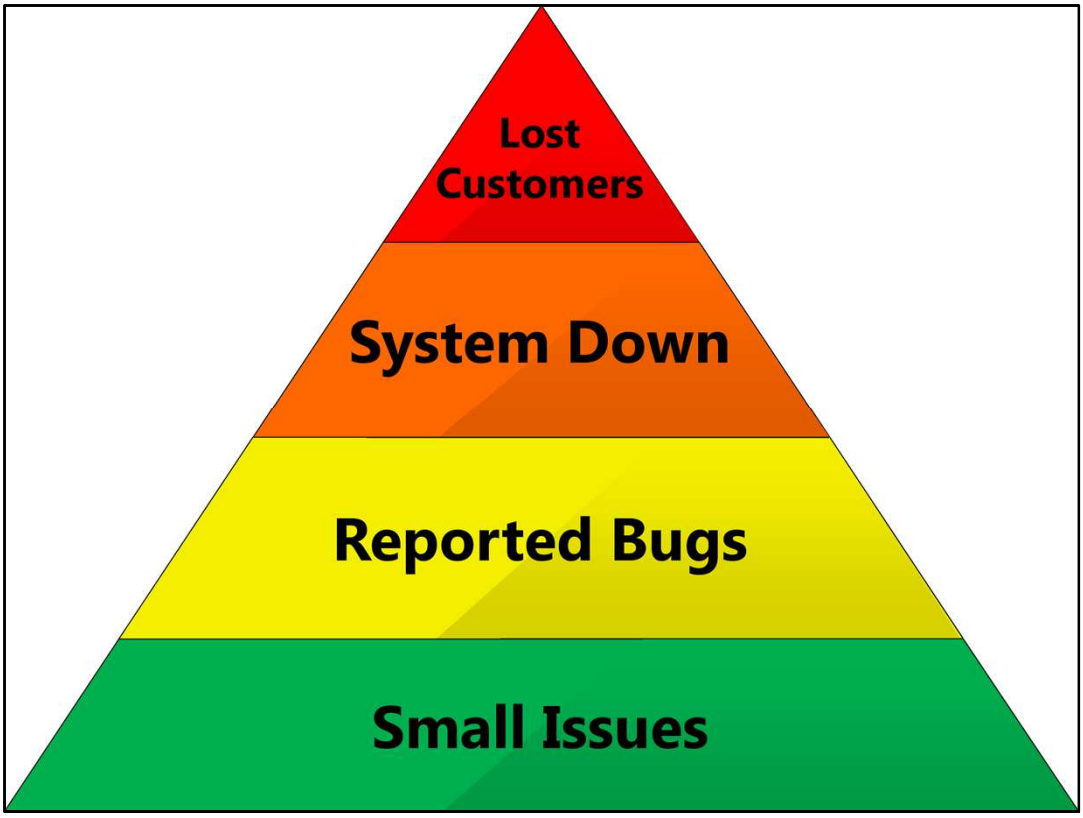
Focus on reducing the bottom layer



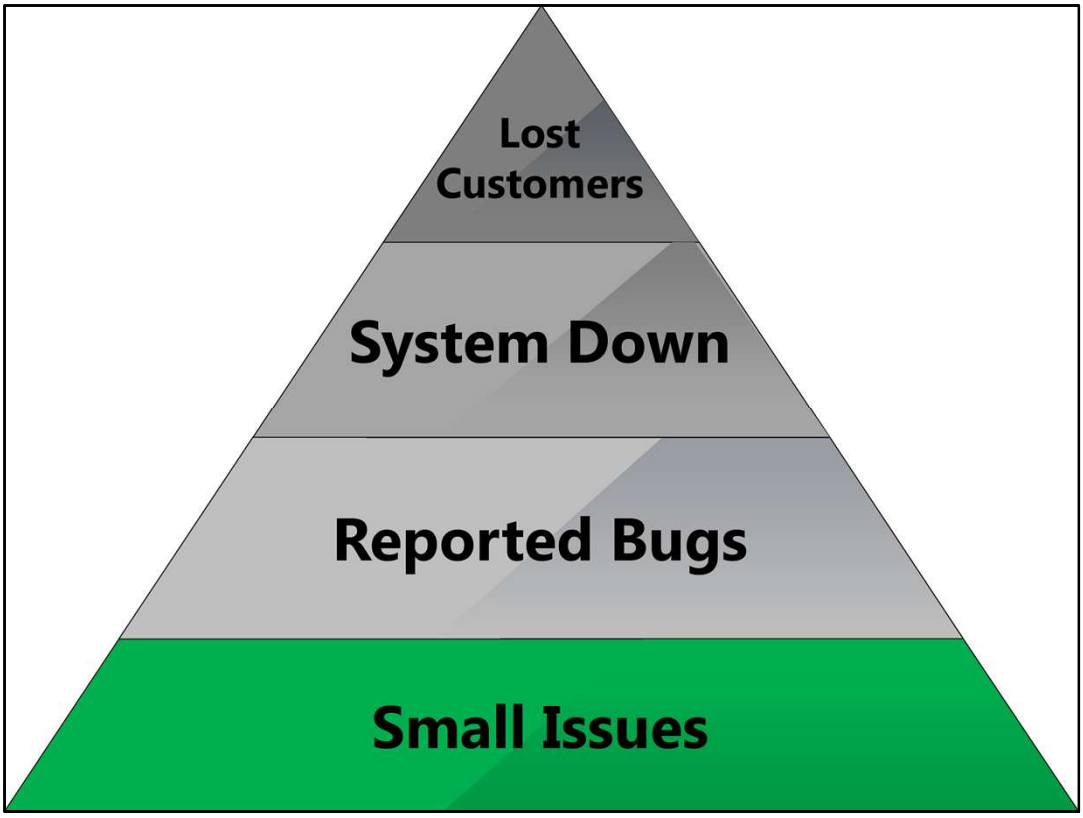
By reducing the bottom layer (i.e. the small mistakes) the end result is that you reduce fatalities



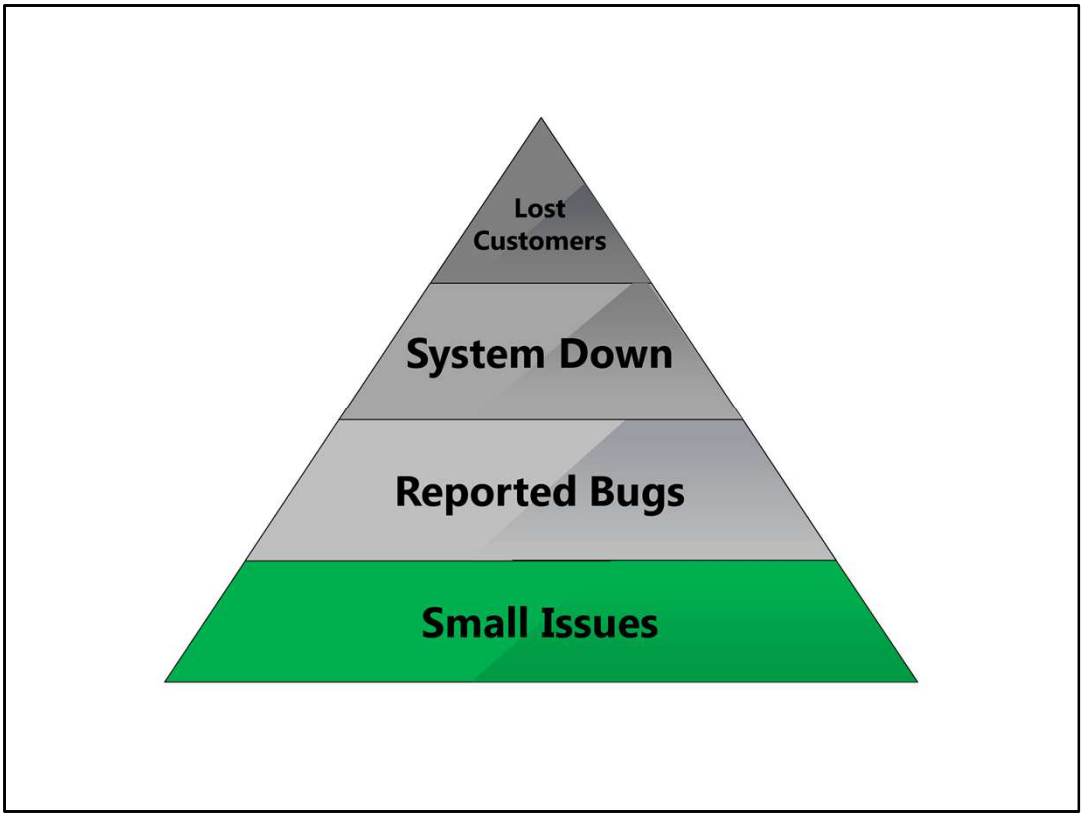
- Story on Oil & Gas pipeline worker
- Summary: Issues usually occur because a small thing was missed or process wasn't followed. Need to focus on those small things



Safety triangle for software development



Focus on reducing the bottom layer



By reducing the bottom layer (i.e. the small mistakes) the end result is that you reduce system downtime and lost costumers.

Implementation

- Get buy in from management
    - Not always easy to do but use business reasons listed in Code Review intro slide
    - Relate reasons in terms of how the business will do better
- Developers
    - Explain the value of code reviews
    - Accountability
        - If person A reviews person B's code, they are as responsible for the outcome of the code
        - Though harsh, this technique avoids people doing a quick scan and passing reviews
    - Have a random round robin approach for reviewing code
        - i.e. junior developers can review senior developers etc
        - Put all names on post-it notes and put on wall. Next time someone needs a review take first person at top of list and that's who reviews your code
    - ALL CODE IS REVIEWED
        - No matter how small

```
...
  l_x := 1;
      l_y := 2;
  IF l_x > l_y THEN
            SELECT nvl(max(e.sal), 0)
INTO l_sal
    FROM    emp e   WHERE
e.empno     = e.empno;
          END IF;
  ...
        EXCEPTION
     WHEN others THEN
NULL;
END;
```

Example: In this case send back right away and format the code properly

What if I wrote
All the text on this
Screen in some odd and
weird
way? It becomes
tough to read.
eh?

If someone says formatting isn't important, show example of plain text writing using the same formatting. It's difficult to read.

```
...
  l_x := 1;
  l_y := 2;
  IF l_x > l_y THEN
    SELECT nvl(max(e.sal), 0)
    INTO l_sal
    FROM emp e
    WHERE e.empno = e.empno;
  END IF;
  ...
EXCEPTION
  WHEN others THEN
    NULL;
END;
```

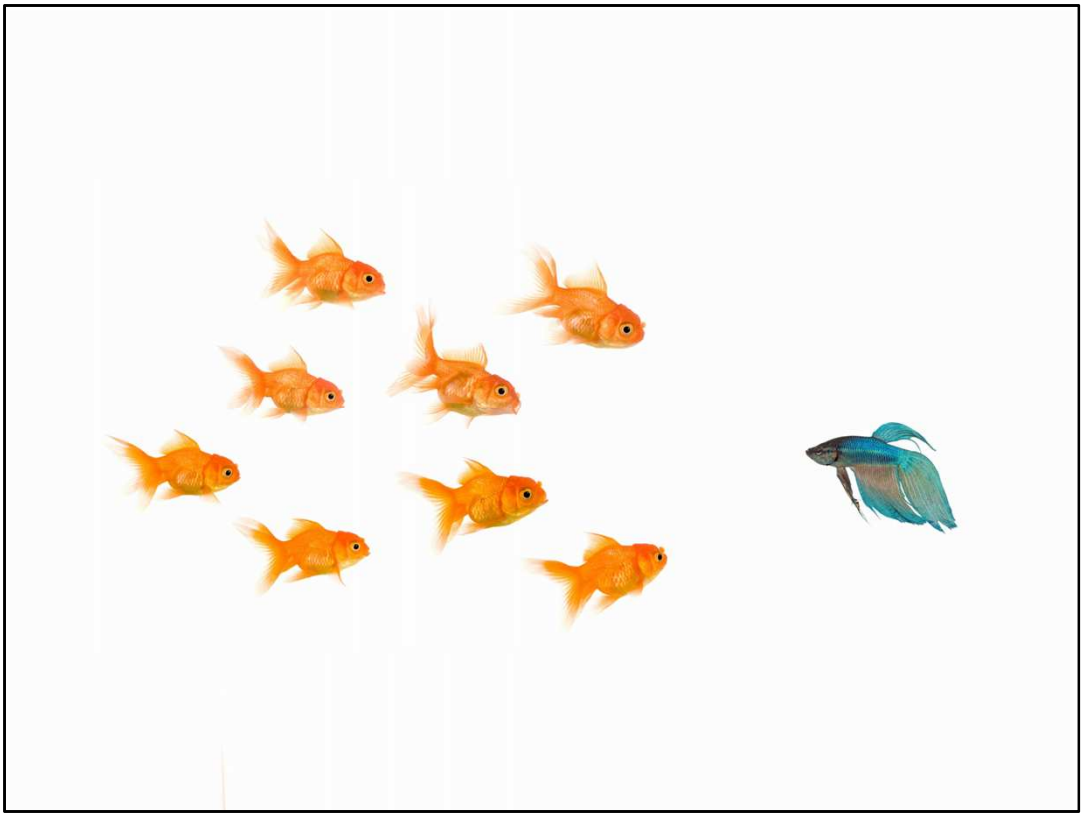
Overall: Indentation is way off, no comments.

100: 1>2 will never occur why have it?

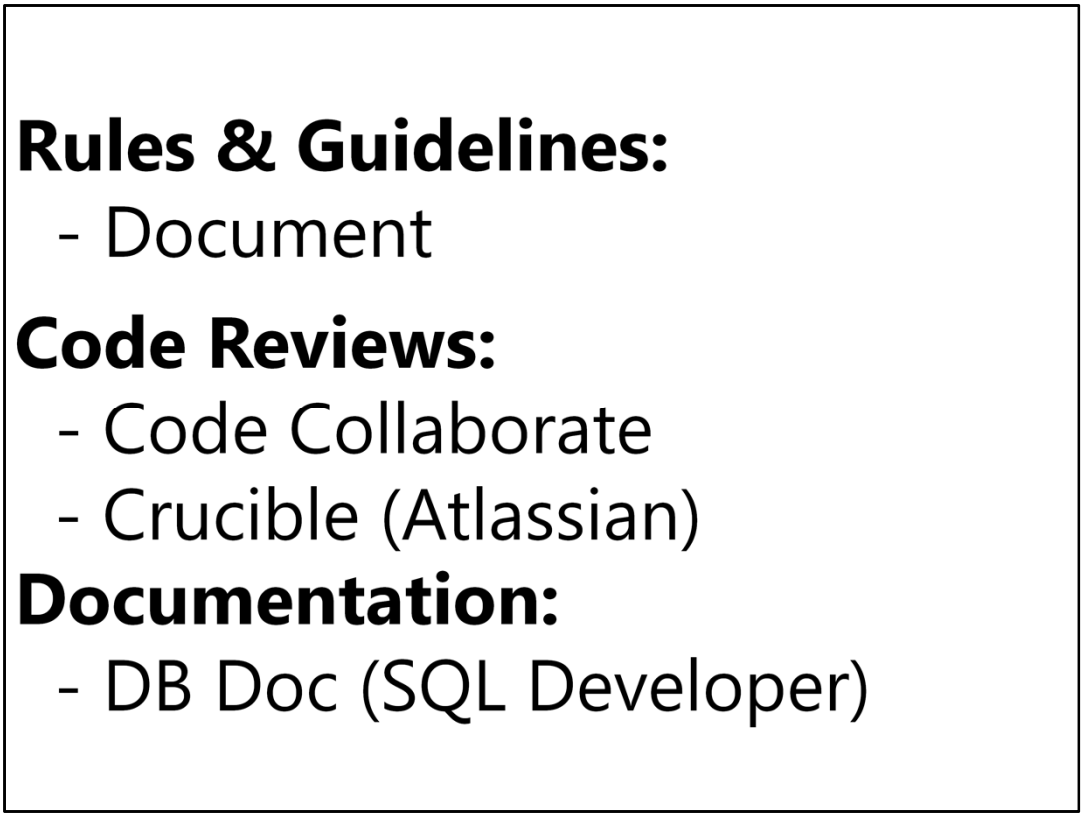
105: e.empno = e.empno.
Shouldn't it be p_emp_no?

130: Why do you have when others then null

Example of what a reviewer would return to the initial developer. Can be done in email form (keep this as simple as possible to start).



You will face challenges, most of them cultural. Stick through it!



Tools:

Code review tools are not required. If initially starting off just use email.

DB Doc for SQL Developer: http://www.thatjeffsmith.com/archive/2012/03/javadoc-for-the-database-a-la-dbdoc-via-sql-developer/


Martin@ClariFit.com


@martindsouza


http://www.TalkApex.com


http://presentations.ClariFit.com

ClariFit

www.ClariFit.com