Taking Ownership and Adapting TSP Successfully Over Time

Intuit Engineering Team

TSP: One Team's Story

- How we've used and adapted TSP over the course of four projects
- Some specific adjustments we've made
- Process changes we're continuing to make

Our History with TSP

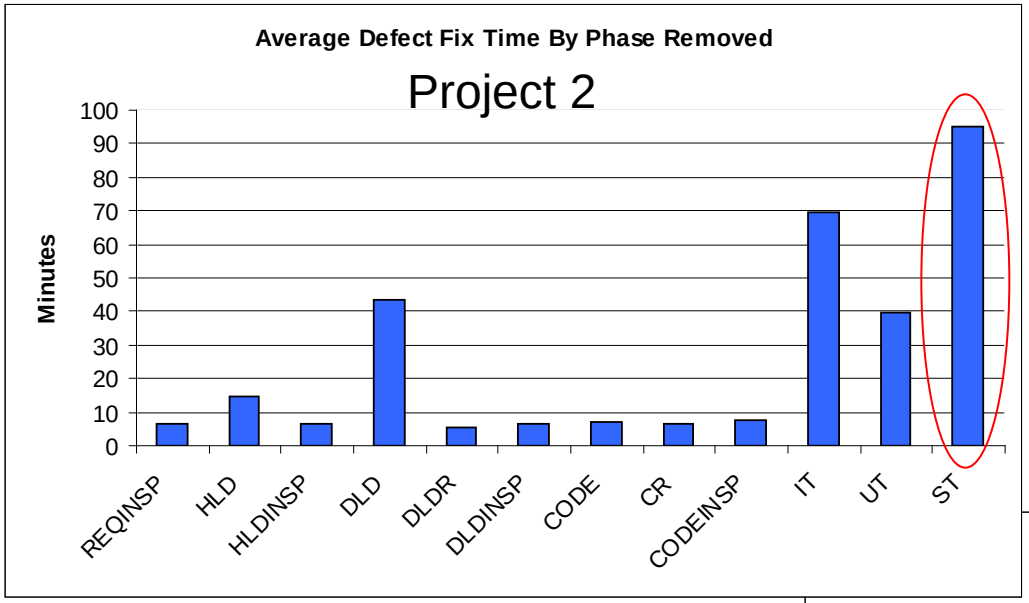
- **Project 0 (2004)**
  - We more or less followed the standard TSP process
- **Project 1 (2005)**
  - Larger team, higher profile project, more challenges
- **Project 2 (2006)**
  - Decided to make some changes based on our experience
- **Project 3 (2007)**
  - Continuing to refine our process

What We're Going to Talk About

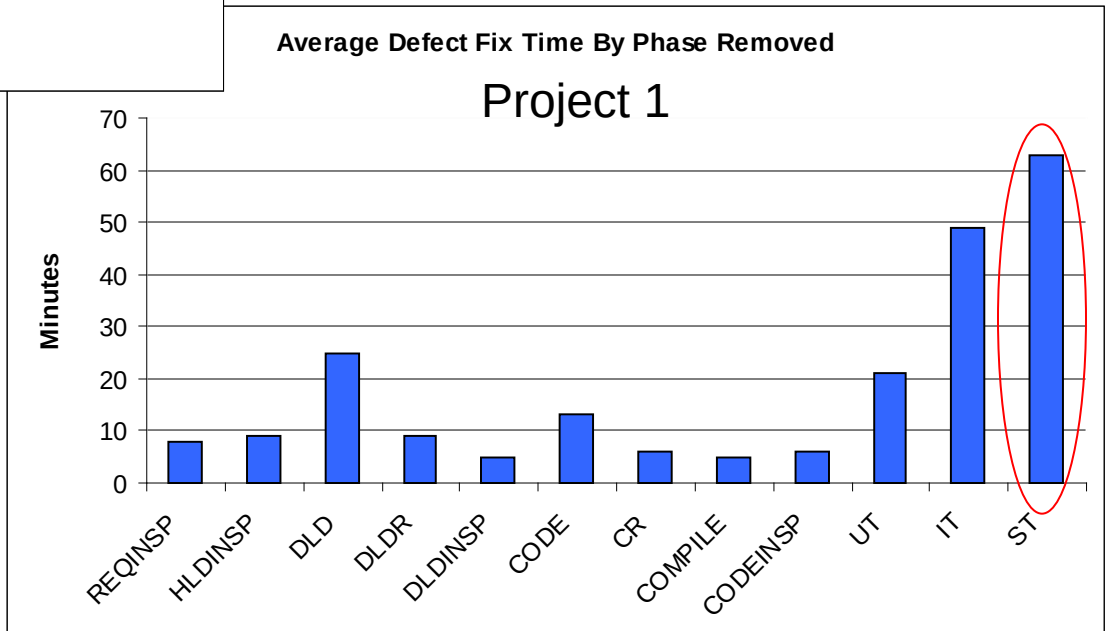
- Reducing System Test (ST) defects
- Improving the requirements process
- Plan accuracy and overall improvements
- Team and management dynamics

Reducing defects found in System Test

Why We Changed



Defects found in system test take the longest to fix



Why We Changed

- **Less defects found by QA is a good thing.**
  - These defects are tracked by senior management and have huge visibility.
  - They also have wider impacts across the organization.
- **When too many defects are found in ST, we may not have time to fix all of them in the way that generates the most customer delight.**

Why We Changed

- **Project 1:**
  - Removed many defects in ST that could have been removed in earlier phases.
  - 46% of the total time spent fixing defects was spent in ST.
  - Time spent in ST was much more than planned for.
    - Plan: 9% of total project effort
    - Actual: 13% of total project effort

What We Did to Change

- **Changed our attitude about defects**
  - We aren’t afraid of finding defects, we welcome finding them in earlier phases.
  - Finding defects earlier rather than later is a good thing.
  - Finding many defects means we did a great job reviewing/inspecting.

What We Did to Change

- **Made process improvements to try to remove defects earlier in the process**
  - Improvements to the requirements process (covered in the next section)
  - Customized and improved checklists for HLD, DLD, and Code Reviews + Inspections
    - Created by the Design Manager and Implementation Manager, reviewed by the team
  - Planned for Integration Testing and did more of it

What Effect it Had

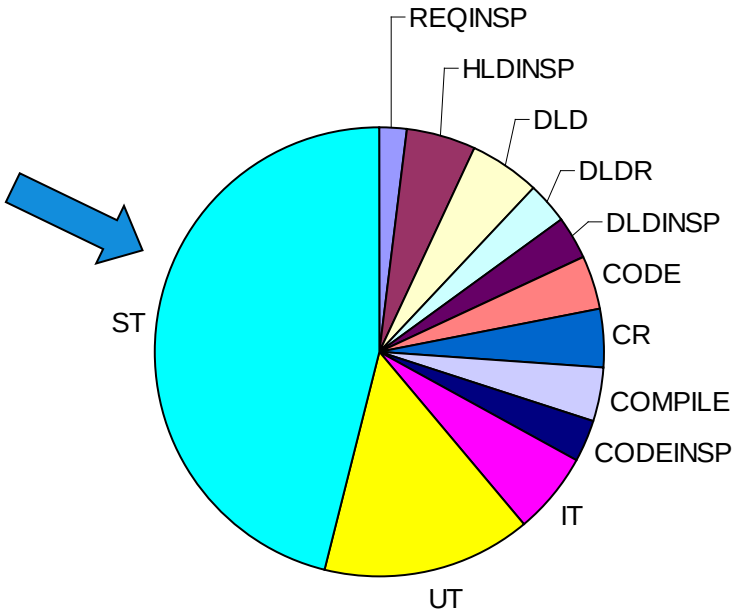
- **Improved phase yields in every defect removal phase where a checklist was used**

Phase	Project 1 Yield	Project 2 Yield
HLD Inspection	39%	55%
DLD Review	19%	21%
DLD Inspection	36%	40%
Code Review	20%	22%
Code Inspection	28%	60%

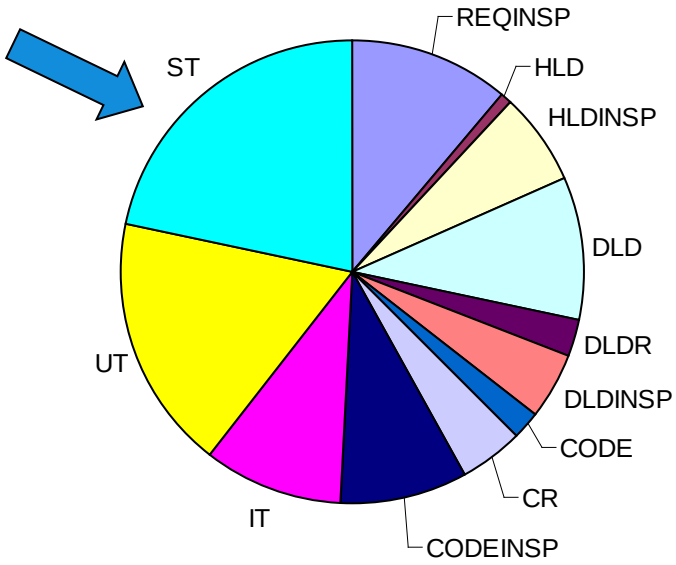
What Effect it Had

- **Smaller percentage of time spent fixing defects was spent in ST**

Project 1:
Total Fix Time by Phase Removed



Project 2:
Total Fix Time by Phase Removed



What Effect it Had

- **Less effort spent in ST**

	% total project hours spent in ST
Project 0	**10%**
Project 1	**13%**
Project 2	**5%**

- **Time spent in ST was very close to plan**
  - Plan: 57.9 hours
  - Actual: 64.7 hours
- **Number of ST defects was lower than plan**
  - Plan: 24.3 defects
  - Actual: 23 defects

How We're Continuing to Evolve

- **More effort towards defect prevention activities:**
  - More collaboration on designs, weekly design office hours
  - Engineers taking ownership of running automated testing, rather than QA
  - Continually improving checklists

Improving the Requirements Process

Requirements – Project 1

- **Engineers “drove” the requirements (cross-functional team brainstormed and answered open questions, but engineers documented design).**
- **Requirements posted on the wiki.**
- **Reviewed mainly by the same people who created the requirements (with addition of a few engineers).**
- **Reviews conducted with a requirements checklist.**

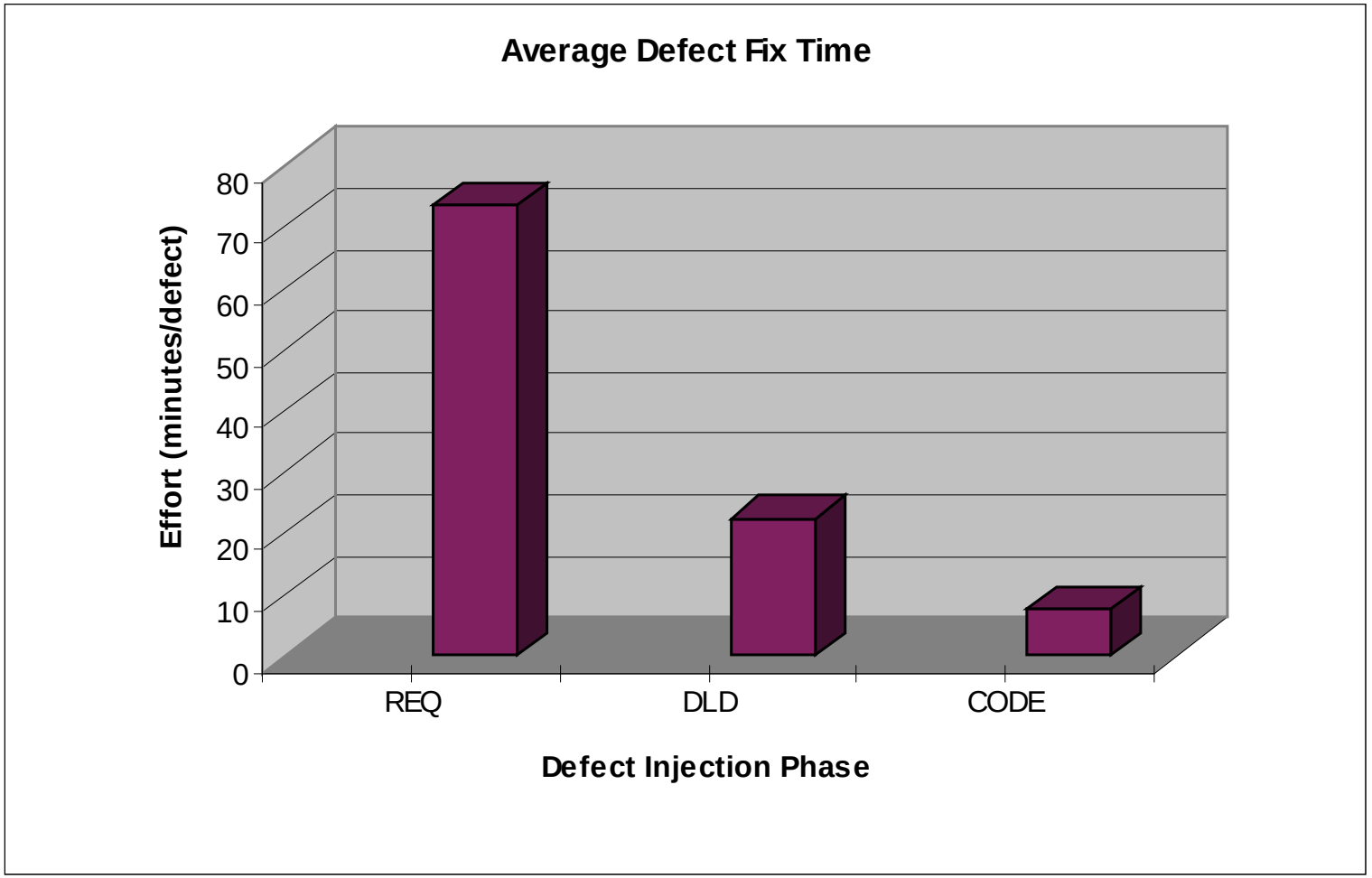
Why We Changed

Project 1 – Post-Mortem Comments

- **"Requirements change is a big deal": Late-changing requirements caused a lot of havoc.**
- **We missed a lot of requirements.**
- **Requirements could have been clearer.**
- **No one had a big picture view of the requirements (people just reviewed pieces).**
- **The requirements specification got stale pretty quickly.**
- **People outside of the team that worked on the requirements should also review them.**
- **"Reviews were great": Formal requirements reviews were extremely helpful.**

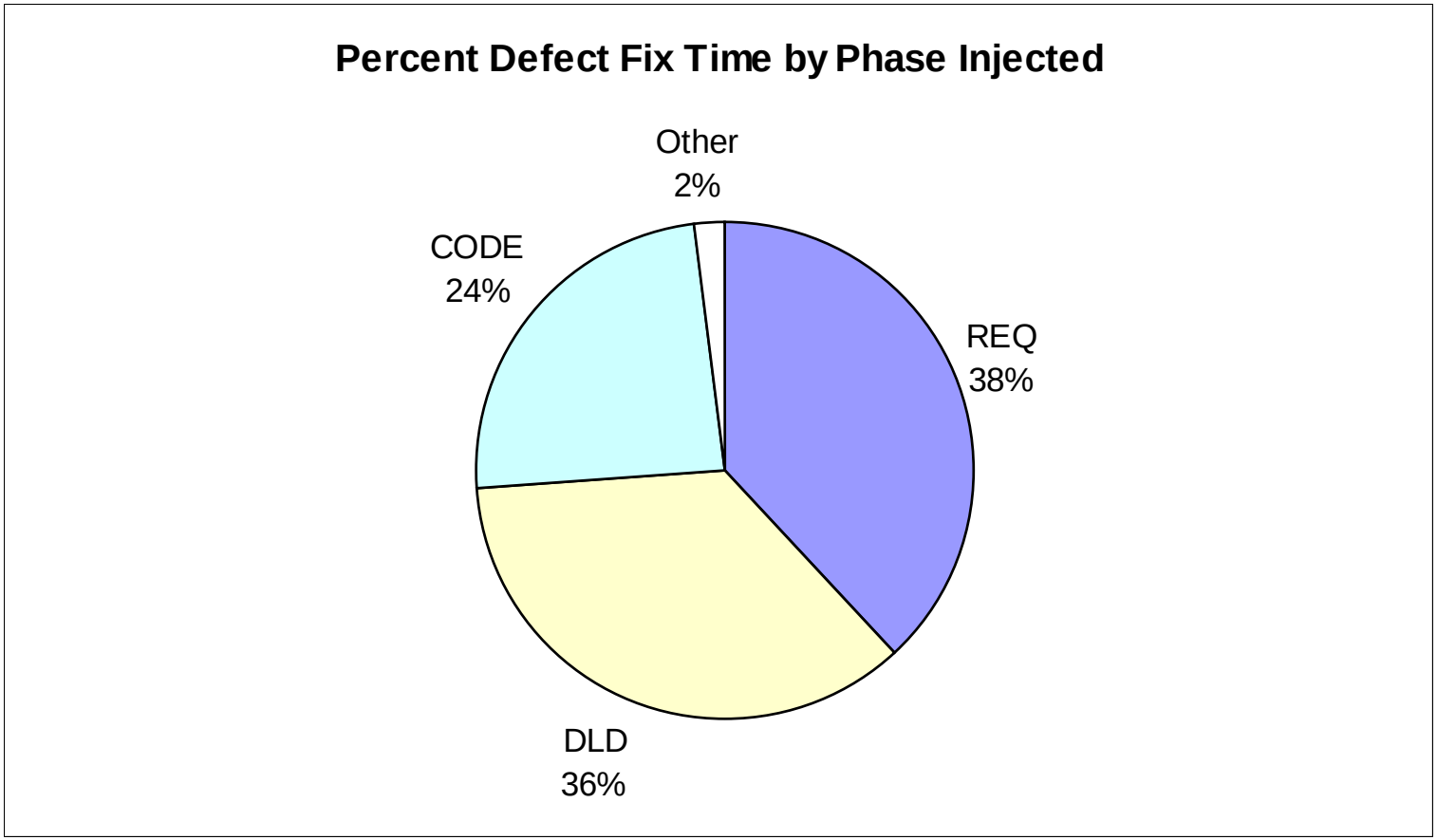
Why We Changed

Project 0 – Average Defect Fix Time by Phase Injected



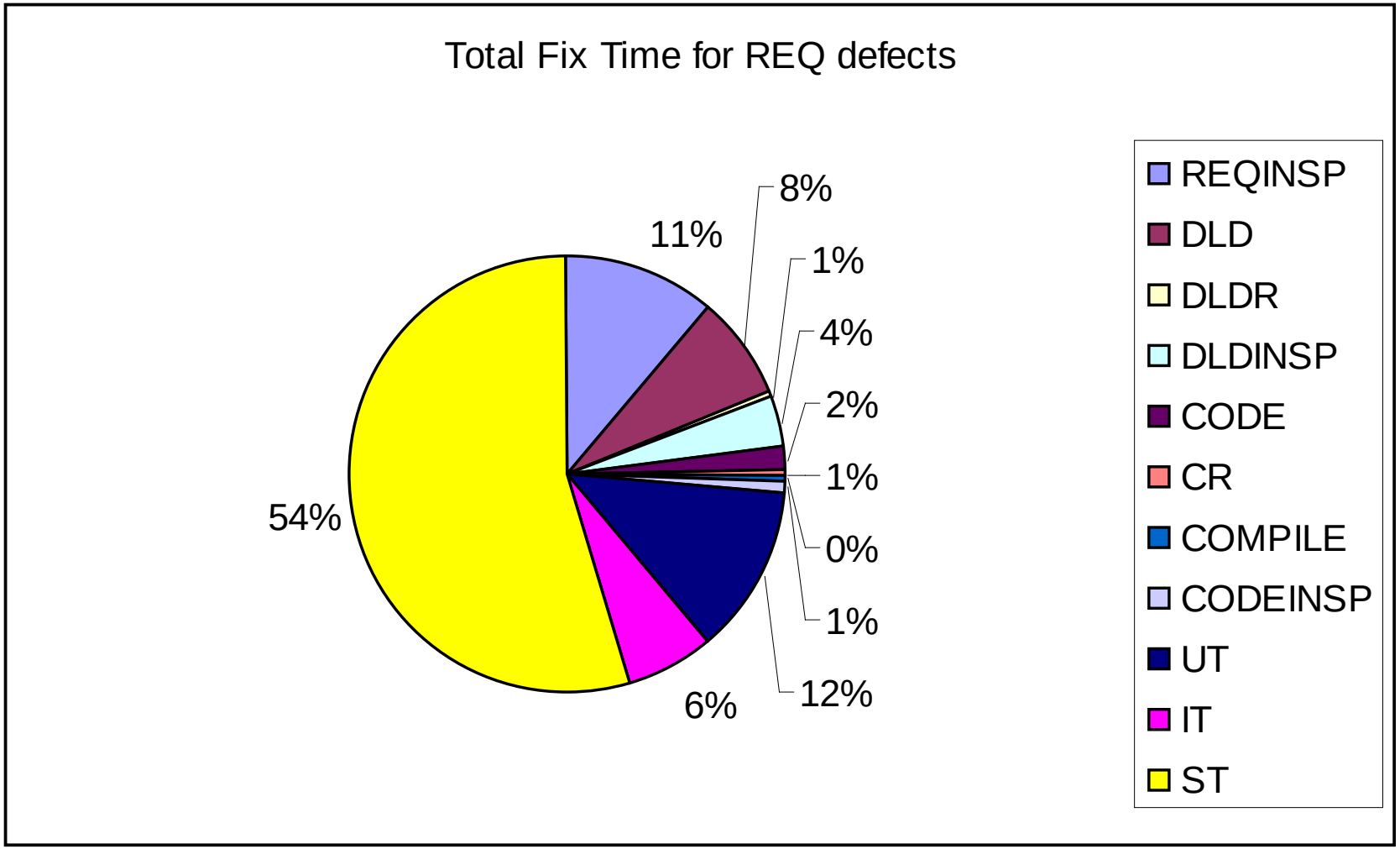
Why We Changed

Project 0 – Percent of Total Defect Fix Time by Phase Injected



Why We Changed

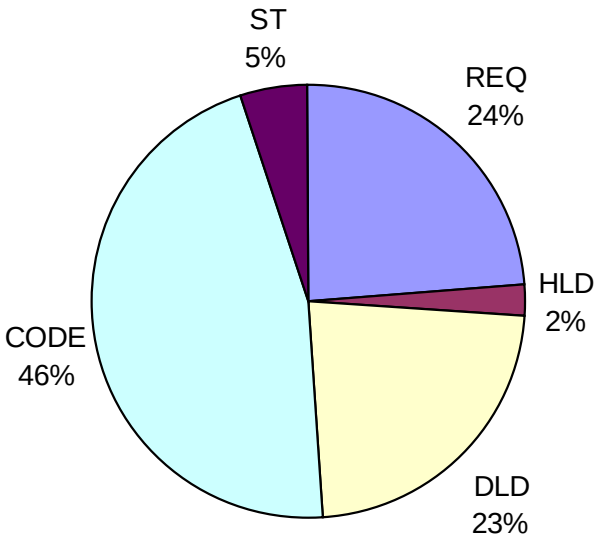
Project 1



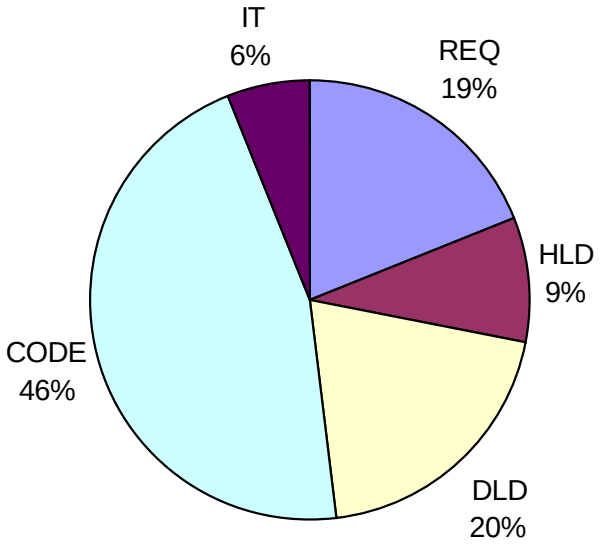
Why We Changed

Project 1 – System Test Defect Data

Injection Phase for ST Defects



Total ST Fix Time by Phase Injected



- **Data is only part of the picture:**
  - Requirements defects weren't tracked consistently
  - Charts only represent bugs **fixed**, not deferred or marked "change not justifiable"
  - We experienced a lot of pain around requirements issues

Summary of Requirements Issues

- Reviews were good and even more people should be involved.
- Requirements defects were not consistently tracked.
- Many requirements were missed.
- Many requirements changed or weren't decided until late in the cycle.
- Defects found in test phases are especially painful to fix.

What We Did to Change

To more consistently track requirements defects/progress:

- **We went from “driving” requirements to “owning” requirements.**
- **Created new defect types around requirements.**
- **Used usability benchmarking to verify the success of our design**

What We Did to Change

To create more complete and correct requirements:

- **High-level and detailed-level requirements, with personal reviews and team inspections for both phases.**
- **Reviews by the whole team, including cross-functional team members.**
- **Created a requirements template based on input from cross-functional team members.**
- **Updated requirements review checklists.**
- **Engineers taking ownership of requirements.**
- **Established requirements “office hours”.**

What We Did to Change

To keep requirements current:

- **Continued using the wiki.**
- **Part of the process of fixing a requirements defect was to update the wiki.**

What Effect it Had

Increased Phase Yield for Requirements Inspection

Phase	Project 1 Yield	Project 2 Yield
REQ Inspection	12%	77%
High-Level Design	0%	3%
HLD Inspection	39%	55%
Detailed Design	9%	9%
DLD Review	19%	21%
DLD Inspection	36%	40%
Code	9%	8%
Code Review	20%	22%
Compile	30%	19%
Code Inspection	28%	60%
Unit Test	58%	55%
Build and Integration Test	23%	38%
System Test	100%	100%

What Effect it Had

Met team usability goal

- **At the end of Project 2, we conducted a user study to see if we met our team usability goal. For the features we made changes to, task completion did indeed improve to be 90% or greater.**

Greater team confidence in requirements

- **Perhaps most importantly, the team felt more confidence in and ownership of its requirements—this was listed as one of the "positives" during our Project 2 postmortem.**

How We're Continuing to Evolve

- Developing use cases
- Working earlier with technical support
- Build in time for exploration of existing functionality
- Blog for better communication

Improving Plan Accuracy

Why We Changed

- Launch process was painful; many hours spent on design and estimating.
- Planning parameters were off and we had to work overtime to make up for it.
- Plan (spreadsheets) stayed static because we didn't feel empowered to change them.
- The plan wasn’t reflecting reality.
- Many key learnings came at the end of the project, when it was too late to make changes.

What We Did to Change

- **Realistic Planning**
  - **Took estimation off-line from launch.**
  - **Estimated at a higher level (Tiny, S, M, L, Freakin' huge).**
  - **Budgeted time for everything - including bug fixing and overhead.**
  - **Used actuals from previous project for a more realistic plan.**
    - **For example, Project 1 data showed more than 1/2 our time spent on design; planned to adjust phase time for the next project.**

What We Did to Change

- **Ownership of individual plans**
  - Modified planning parameters for individual spreadsheet.
  - Revisited and revised estimates at any time during the process.
- **Re-evaluate often**
  - Re-launched after requirements phase when more was known.
  - Post-mortem after each phase. Made changes to process mid-cycle.
  - Weekly meeting used as a process improvement/refinement tool.
  - Weekly review of team goals and risks helped keep awareness; for example, discussed impact of new requirements.

What Effect it Had

- **Realistic Planning**
  - Finished on time. Available to help other teams.
  - Reduced scope early in the release when the decision is less painful, rather than waiting until a lot of work has already been done.
- **Ownership of individual plans**
  - Good work/life balance.
- **Re-evaluate often**
  - Real-time changes to our process. Process change used for next phase and in place for next project.
  - Got early start for next year.

What Effect it Had – Project Data

Project 1

- **Plan v. Actual**
  - Sizes (LOC): 46% growth of plan
  - Effort (hours): 18% growth
- **Taking into account de-scoping, size growth was actually 104%!**
- **Underestimation was the trend.**

Project 2

- **Plan v. Actual**
  - Sizes (LOC): overestimated by 18%
  - Effort (hours): shrank by 24%
- **Reduction due to early reduction of scope.**
- **Tracked data for overhead.**
  - Number of bugs fixed
  - Average time to fix 1 bug

How We're Continuing to Evolve

- **More off-line preparation for launches.**
- **Revise and refine estimates, taking into account**
  - Data from previous projects
  - Code we've worked in before
- **Continue to evaluate how process is working.**
  - Weekly team meetings
  - Re-launches
  - Post-mortems

Other Process Improvements

- **We ‘defused’ some common TSP concerns.**
- **Managers help, rather than hinder, our progress.**
- **Team attitude has helped a lot.**

We Defused TSP Concerns

- **Fear: Could evil outsiders monitor individual team member performance? Just how long did you spend at lunch?**
- **Solution: Use alternate names. Brady Bunch characters, Sesame Street characters, other ways of obfuscating data.**

We Defused TSP Concerns

- **Fear: ‘Gasp factor’ when defects are found and reported.**
- **Solution: Changing team attitude about defects. Rather than a measure of POOR workmanship (i.e. injecting a bug is bad), finding bugs is a measure of GOOD work.**

We Defused TSP Concerns

- **Fear: Productivity of individuals compared by management (e.g. “Grover works faster than Big Bird - why?")**
- **Solution: Team discusses overall productivity proactively to manage expectations (e.g. include time for non-project work, vacations, and other activities). Individual concerns can be discussed one-on-one with coach privately. No individual spreadsheets are shown in the weekly team meeting; we look at the rollup for the whole team.**

How Management Can Help

- Our team has stayed largely intact over multiple years.
- TSP one-on-ones with coach provide for consistency in tracking and other project issues.
- Coach is internal and has been with the team for several projects.
- Manager one-on-ones therefore allow for time to talk about non-project issues.
- Team goals and risks are owned by the whole team, managers included.
- Whole team (including managers) decides what is being delivered. Managers are supportive, not dictatorial.

Team Attitude Counts

- **Being able to adapt the process to our needs has made our lives a lot more pleasant.**
- **We seek to improve our processes to make us more efficient and further reduce ST bugs.**
- **Each team member is also a leader, and we jointly make decisions.**
- **Careful planning helps maintain a good work-life balance.**
- **Engineers take ownership of requirements; if the requirements aren't good, that now reflects on US.**
- **We all use the blog and wiki to capture ideas; there is no one scribe.**
- **Everyone is willing to try things; we are good at compromising and cooperating.**

Summary

- **Data and team interest help us decide what phases to focus improvement on.**
- **Data helps us assess the value of our improvements.**
- **Ultimately, we do what works.**
- **We own our team process.**

Q & A