QAI /QAAM 2011 Conference
“Proven Practices For Managing
and Testing IT Projects”

Defect Analysis: The Foundation of Process Improvement

David E.Oddis

10th Annual Mid-Atlantic
Software Quality and Program
Management Conference

9/27/2011

Defect Analysis: The Foundation of Process Improvement

David Oddis
Swathi Gabbita

Introduction

Case Method Analysis

How we performed defect analysis to indentify focus areas for process improvement.

Background

Background information

- Project background
- Intro to the problem
  - Quality issues

Goals of the presentation

- Walk through of a real world defect analysis technique
- Provide the audience with a repeatable, proven process and template to take back to their organization

It's OUR workshop – please participate and share your lessons learned, challenges, approaches, & ideas.

Agenda

- Goals & Objectives
- Approach
- Scope & Team
- Ground Rules for Analysis
- Clusters, Mitigation Strategies
- Findings
  - Rollup by Category
  - Breakdown in Implementation Categories
  - Rollup by Issue
- Top Areas of Focus
- Some Inferences
- Themes of Opportunities
- Recommendations

Goals & Objectives

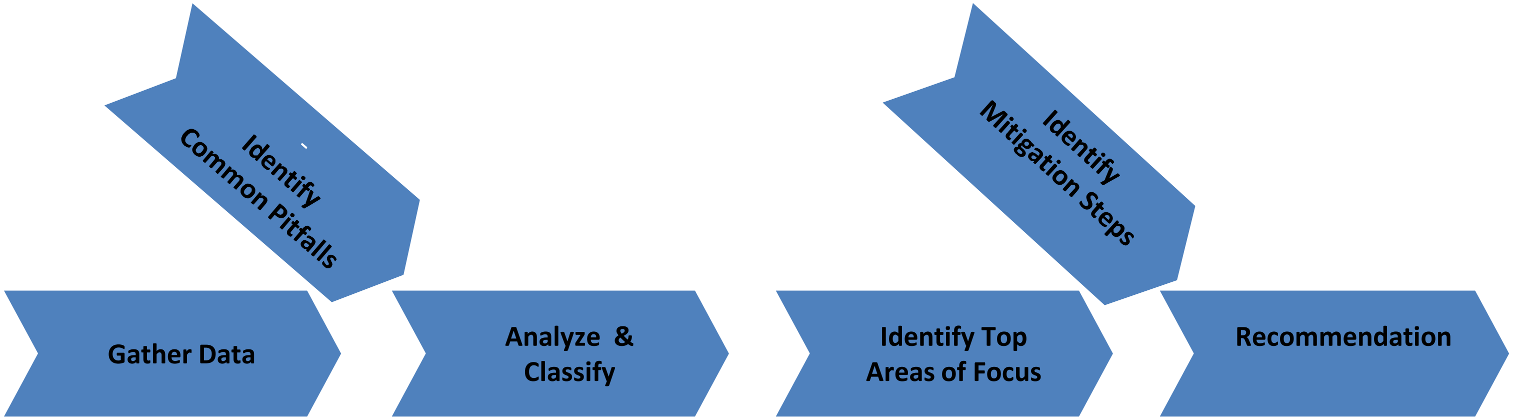
Goal

- Identify areas and earliest opportunities to enable the development of high quality software, as a collaborative effort between Development and QA teams

Objectives

- Analyze, Retrospectively, the actual data gathered on {n}project
- Identify the top {n} areas for improvement
- Recommend action plans

Approach



- ✓Refine the Common pitfalls with more insight from analysis
- ✓Pilot the approach with a small sample
- ✓ Validate the insights
- ✓ Improve and repeat on the rest of the areas

Scope & Team

Scope

- A fairly self contained data sample of a representative workload under the challenges and constraints of {x} project
- 100 out of 1100 random defect tickets

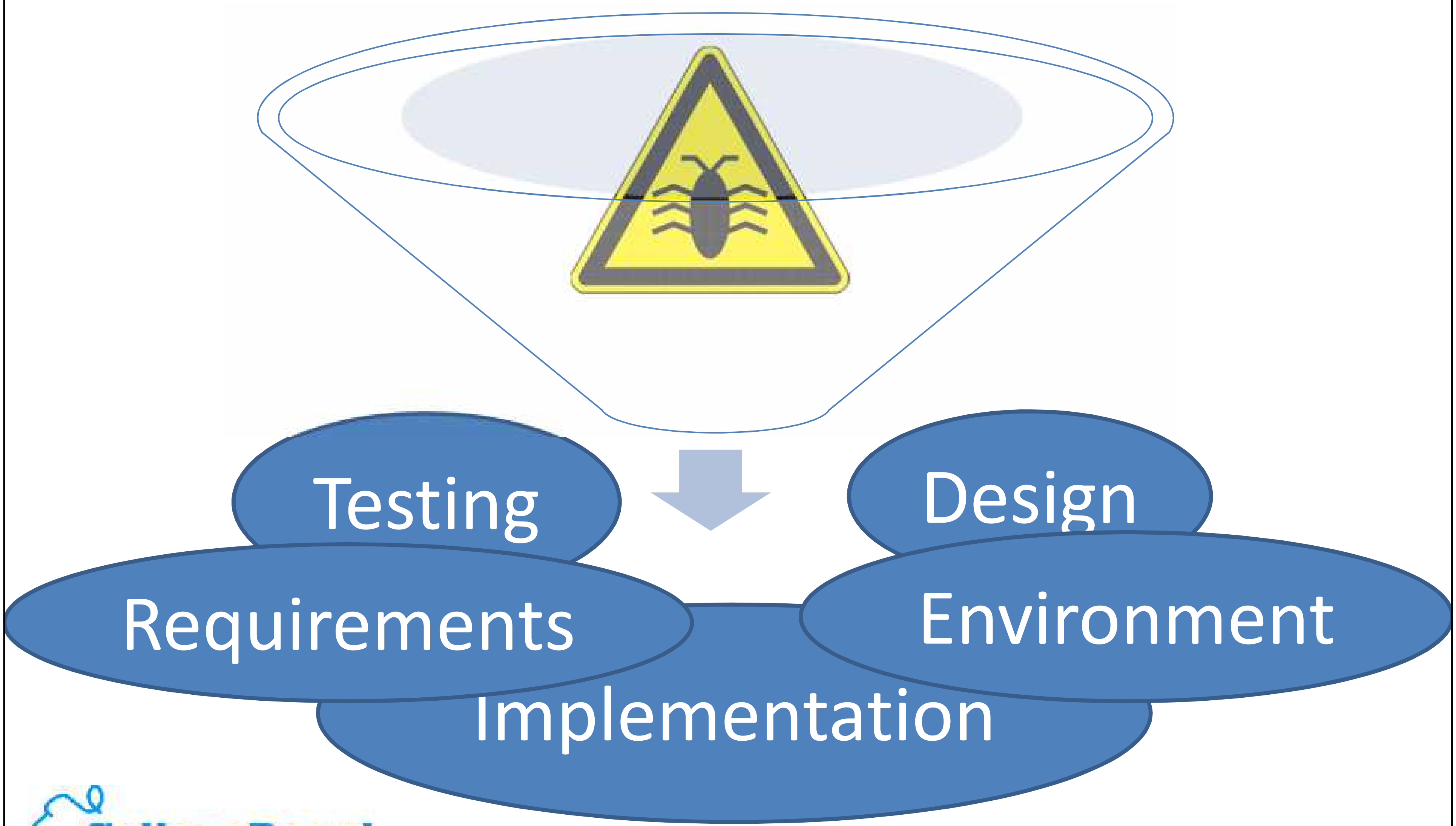
Team

- Leaders – Contribute, Moderate and Recommend
- Working Team – Analyze and contribute to the recommendation
  - ➢ Development
  - ➢ Quality Assurance
  - ➢ Data Base Administrator
  - ➢ Architect

Ground Rules for the Analysis

- Be open, bold and honest
- Do not defend
- Be objective
- Be open-minded
- Focus on the data and facts
- No opinions on behalf of others
- Emphasis on the issue, not on the party

Clusters (Common pitfalls)



Testing

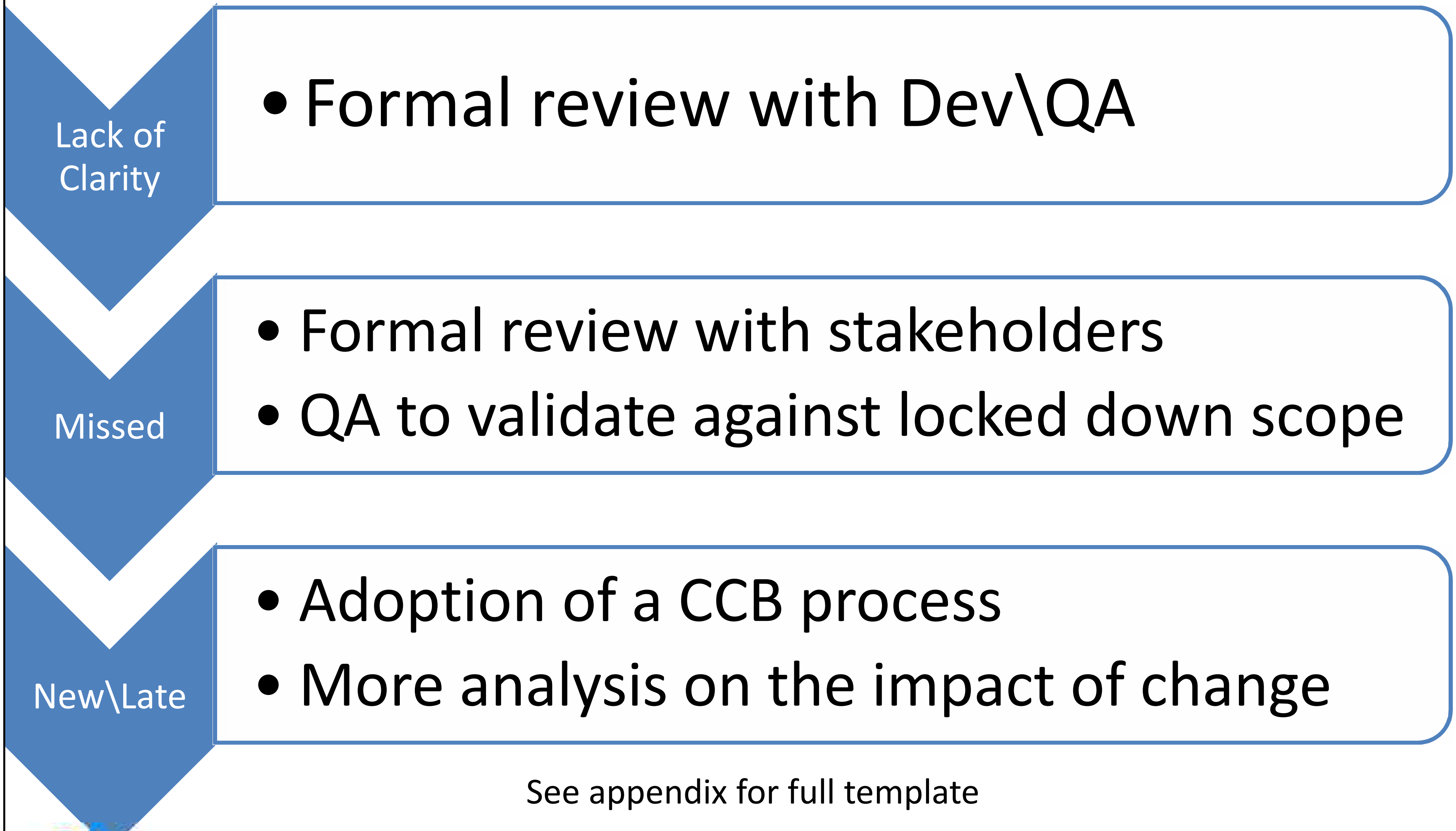
Design

Requirements

Environment

Implementation

Requirements (Common pitfalls & Mitigation)



Lack of Clarity

- Formal review with Dev\QA

Missed

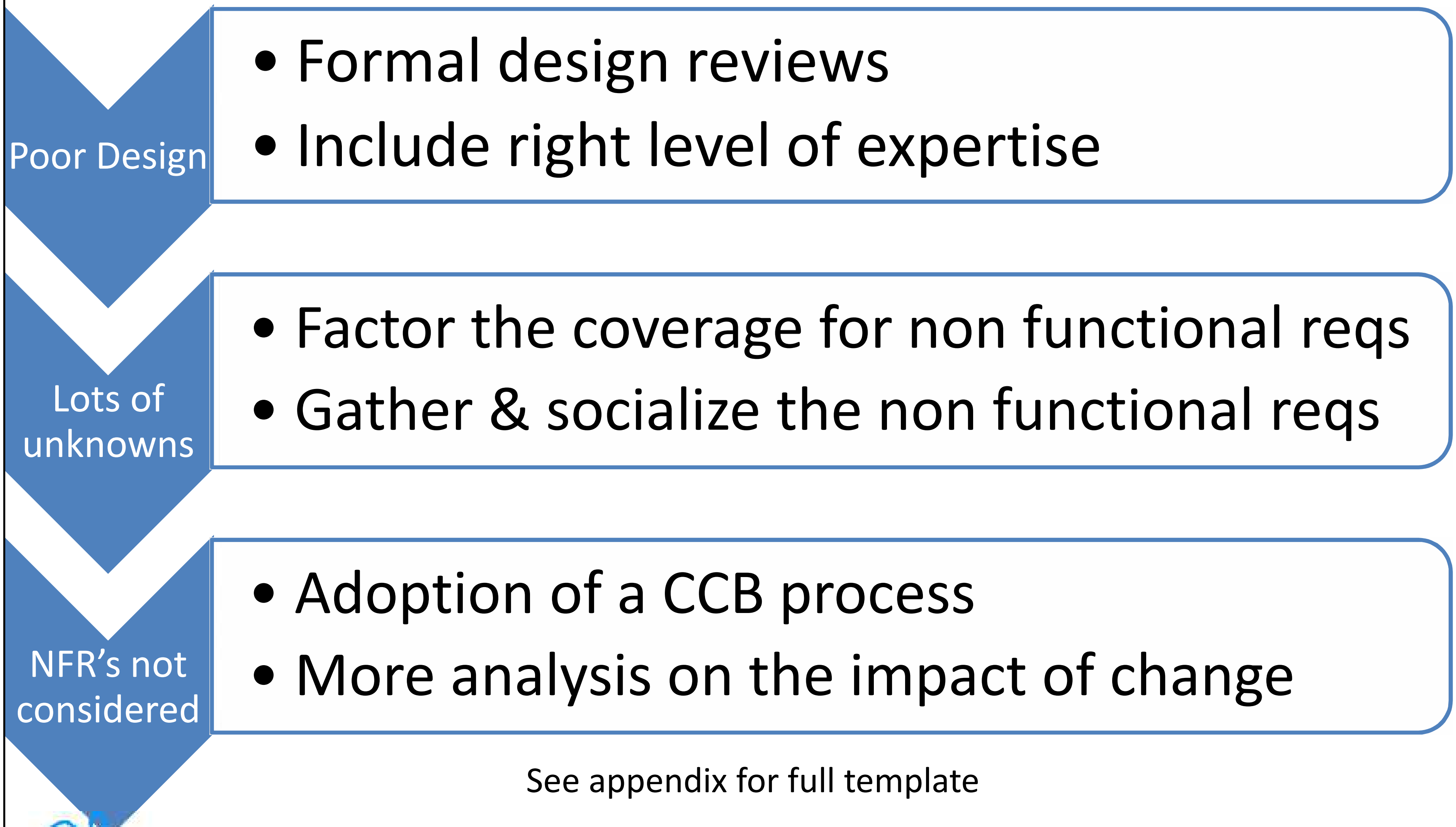
- Formal review with stakeholders
- QA to validate against locked down scope

New\Late

- Adoption of a CCB process
- More analysis on the impact of change

See appendix for full template

Design (Common pitfalls & Mitigation)



Poor Design

- Formal design reviews
- Include right level of expertise

Lots of unknowns

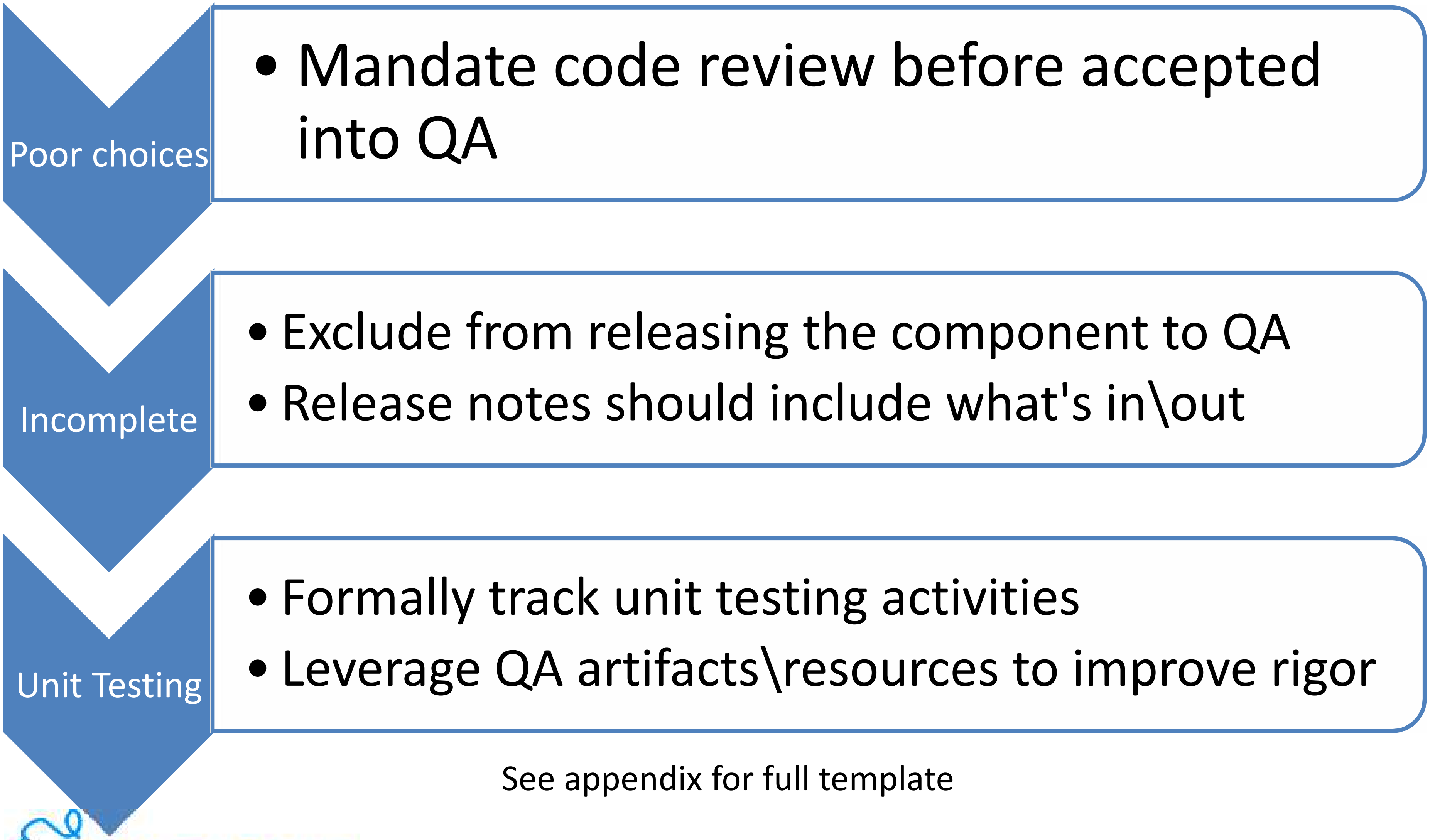
- Factor the coverage for non functional reqs
- Gather & socialize the non functional reqs

NFR's not considered

- Adoption of a CCB process
- More analysis on the impact of change

See appendix for full template

Implementation (Common pitfalls & Mitigation)



Poor choices

- Mandate code review before accepted into QA

Incomplete

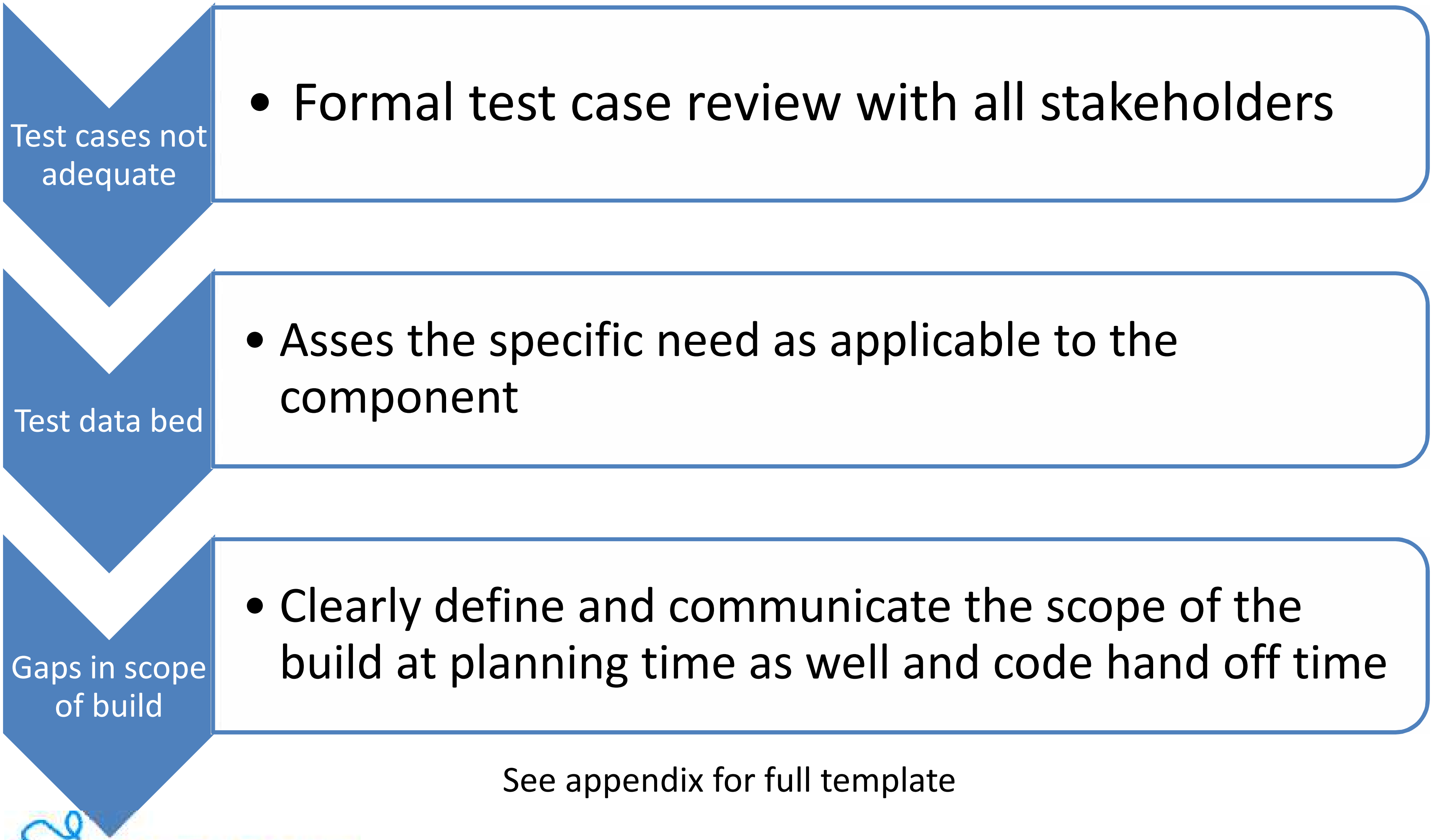
- Exclude from releasing the component to QA
- Release notes should include what's in\out

Unit Testing

- Formally track unit testing activities
- Leverage QA artifacts\resources to improve rigor

See appendix for full template

Testing (Common pitfalls & Mitigation)



Test cases not adequate

- Formal test case review with all stakeholders

Test data bed

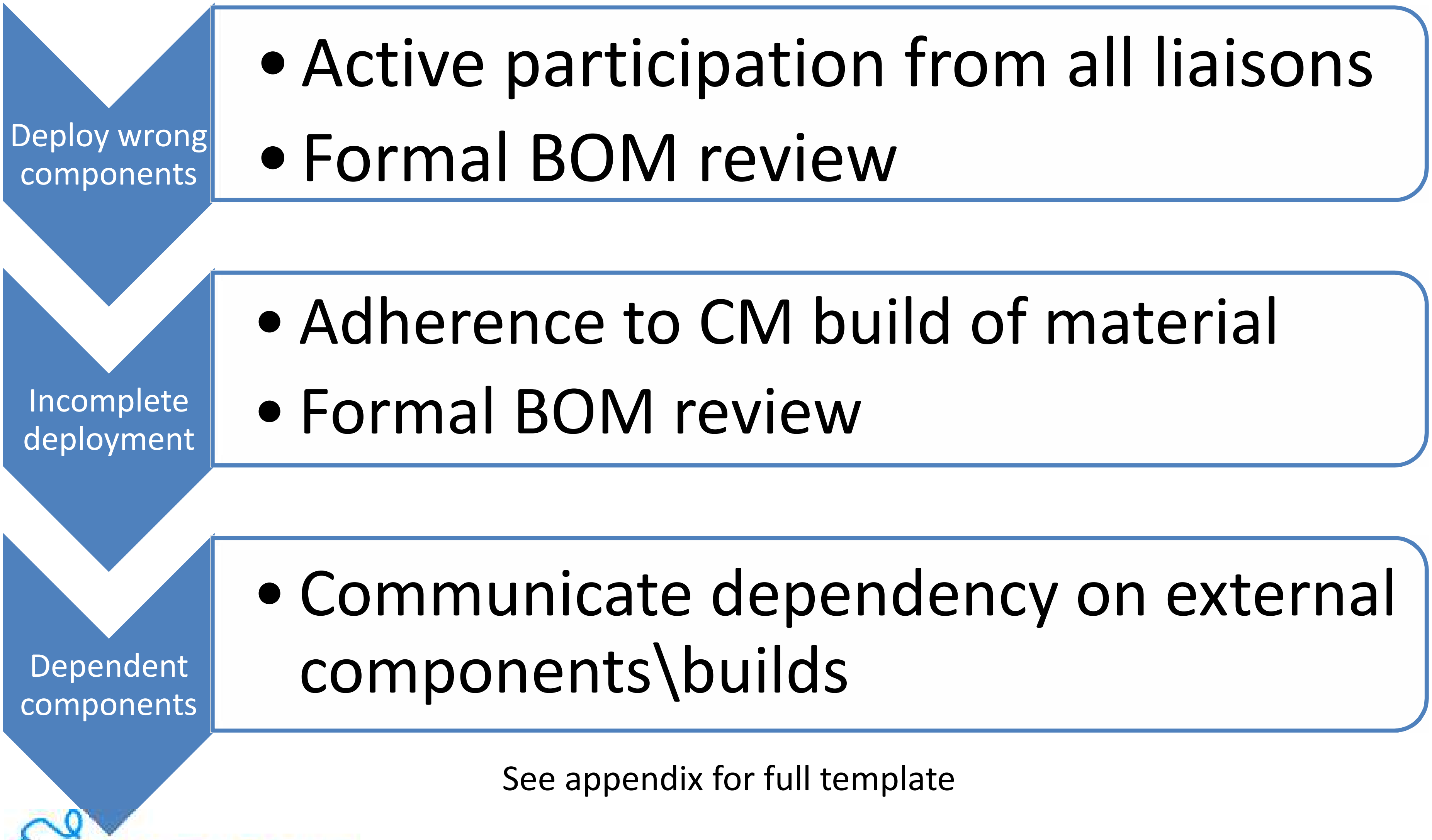
- Asses the specific need as applicable to the component

Gaps in scope of build

- Clearly define and communicate the scope of the build at planning time as well and code hand off time

See appendix for full template

Environment(Common pitfalls & Mitigation)



Deploy wrong components

- Active participation from all liaisons
- Formal BOM review

Incomplete deployment

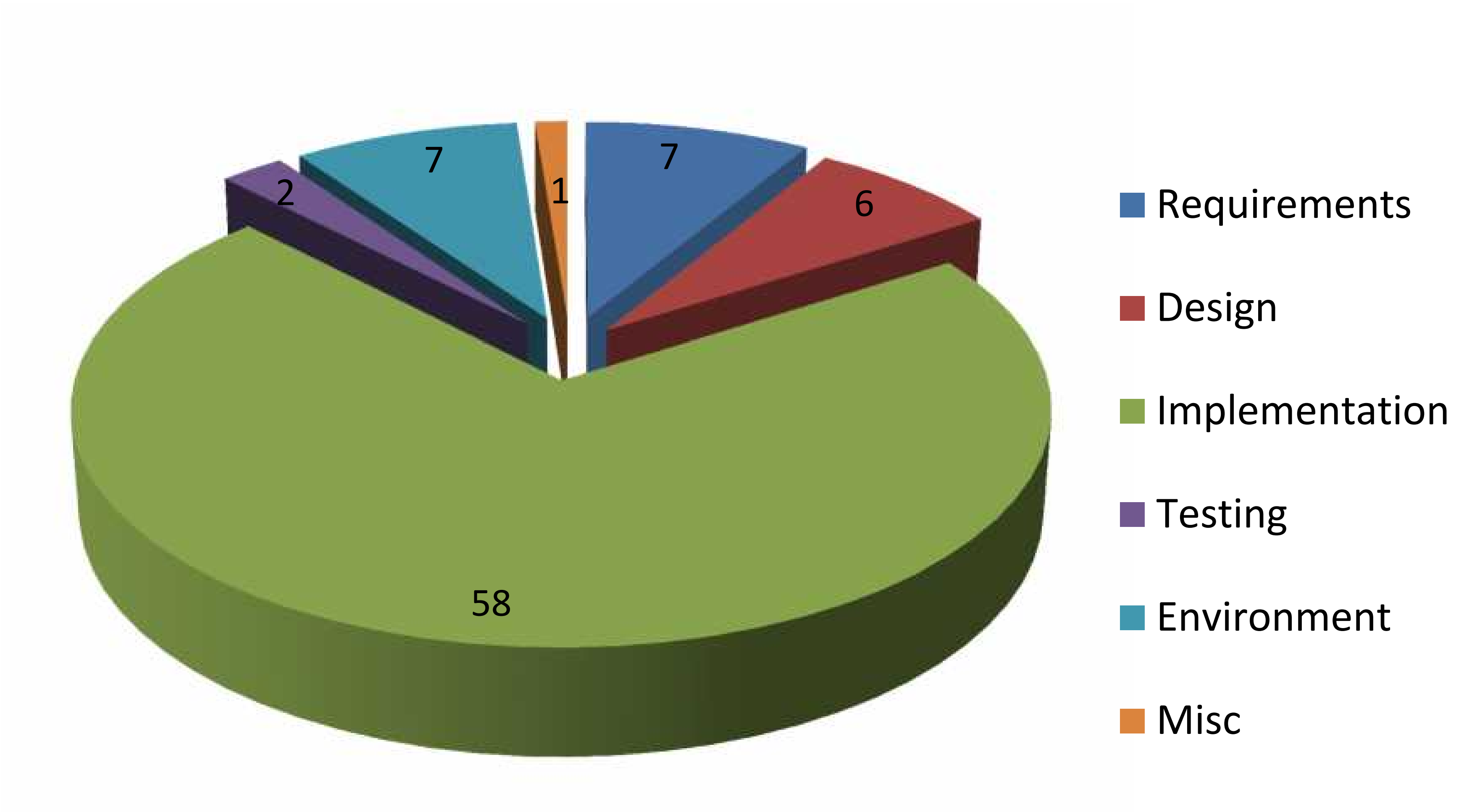
- Adherence to CM build of material
- Formal BOM review

Dependent components

- Communicate dependency on external components\builds

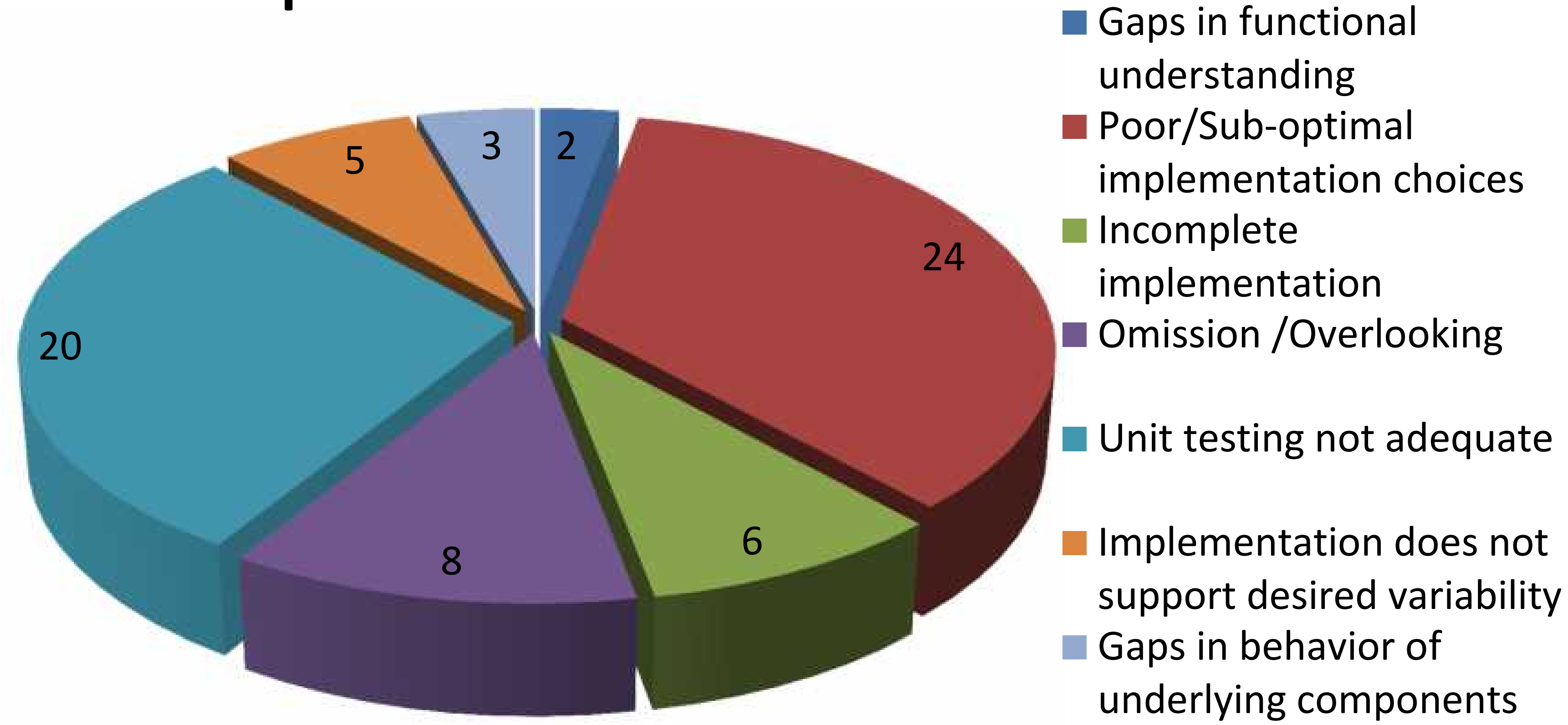
See appendix for full template

Findings – Rollup by Category



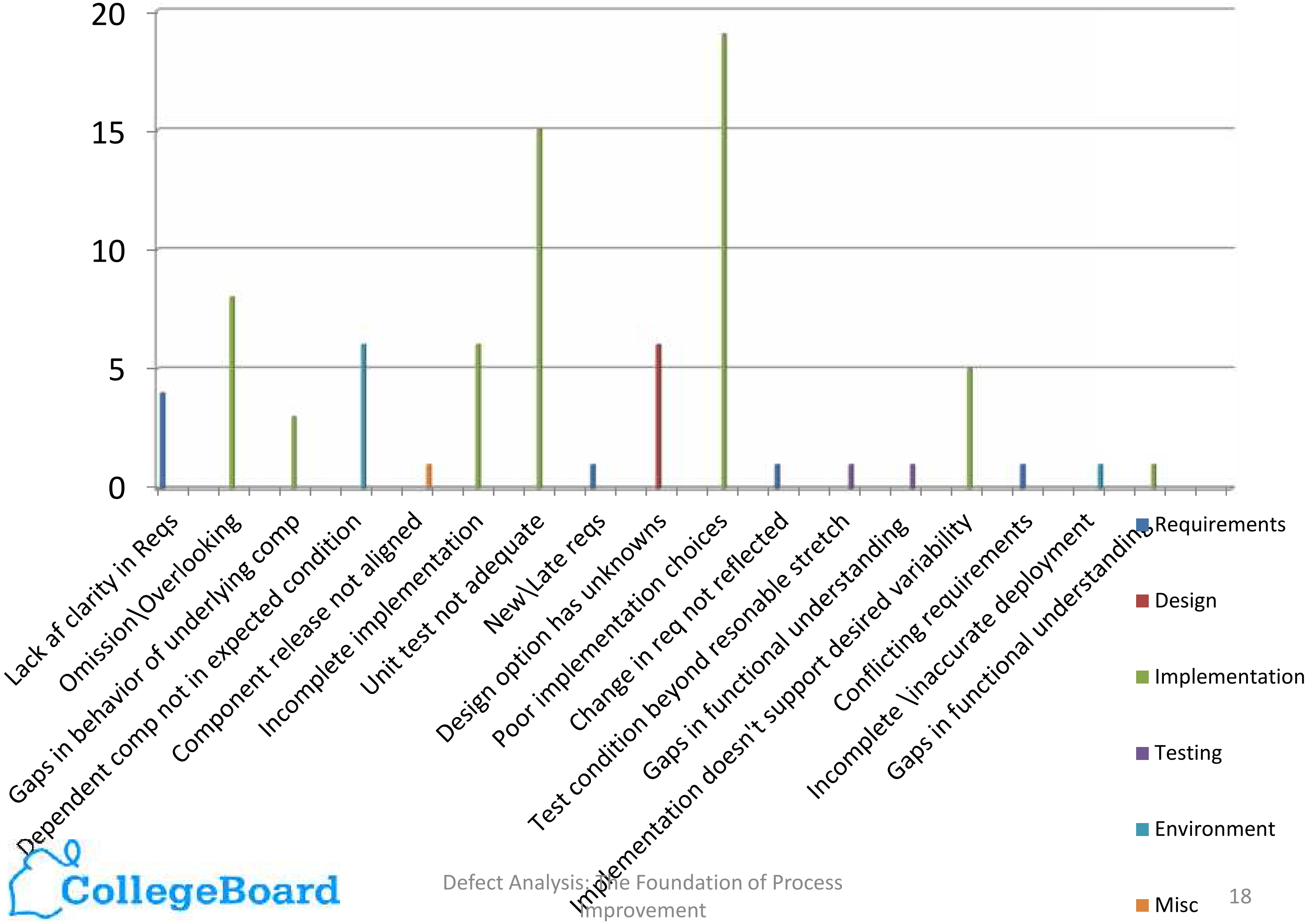
Issues falling into multiple areas is counted into each of them

Breakdown of issues in implementation



Items listed under unit testing are also factored into other categories. Number represents the issues that could be detected even with minimal validation (hard to miss)

Rollup by Issue



Top Areas of Focus

- Poor/Sub-optimal Implementation Choices
- Unit Testing is not adequate
- Omission/Overlooking during implementation
- Incomplete Implementation
- Unknowns with the design options
- Lack of clarity in requirements
- Gaps in the behavior of the underlying component

Some Inferences

Requirements not locked down in time

Requirement changes/clarifications are trickling quite late (from business reviews, deduction of rules..)

Negotiations on complex requirements lasted long

Missed opportunities in the Implementation approach

Insufficient due diligence on the design features with respect to the unknowns supporting the complex presentation requirements

Implementation with unknowns in design, is explored on the actual deliverable

Developer level involvement during prototypes to capitalize on the leanings? (security, Customization…)

Code base is open, for the sample being analyzed, from 4/1 through 6/20

Features were implemented across builds thereby keeping the code active (Security filters later…)

Far too long into the cycle, team tried to accommodate functionality that doesn't quite fit in

Fluidity in requirements

Gaps in Planning, Communication and Management

Lack of formal quality controls and tracking it from upstream (reviews, checkpoints..)

Incomplete implementations suggests plans were overly aggressive (at the least in the beginning)

Lack of build readiness checks on pre-scheduled builds

Incompleteness of build/implementation is not communicated well

Alignment of deliverable across tracks wasn't adequate

Themes of Opportunities

- **More emphasis on planning. Specific areas of focus include**
  - The Fluidity of the requirements
  - Unknowns with the design option
  - Cost/Benefit of frequency of the builds
  - Strengths and weakness of the members
  - Ramp-up of resources on projects
  - Due Diligence on continued alignment across tracks

- **Incorporation of quality gates into the SDLC. Specifically,**
  - Requirements lock down, review
  - High and Low level designs, reviews
  - Code reviews
  - Rigor in unit testing

Themes of Opportunities

- **Increased collaboration**
  - QA to assist DEV on unit testing resources/efforts earlier in the life cycle
  - QA assistance in validation of requirements against scope, clarity, conflicts etc…
  - Communication on the actual state of builds (cross tracks and disciplines)
  - Timely and effective negotiations with business/business partners on the functional details

- **Accountability and Recognition for maintaining high quality levels**
  - Establish clear sense of ownership and accountability

Recommendations

Based off the quantitative data from the analysis and general inferences drawn, the following recommendations are being made

Planning & Management

•More rigorous program level plan that reflects and actively tracks, at the required level of details, the build & rollout dependencies across tracks

•Incorporation of quality gates into the plan and enforcing them with rigor

•Identify & plan the environment/ data/other component needs, at the planning stages (in contrast to the trade offs)

•Leverage QA artifacts, resources to improve the quality of developer testing

•Assess the readiness of the build itself based on the state of the component. If it's vital to continue with an incomplete component, release/build notes should include a clear articulation of what's in and out

•Factor in and provide the feedback on the quality levels, delivered by an individual, during project close out, performance reviews and/or at the right time as appropriate

•More emphasis on the build/release notes and effectively using them in readiness assessments

Recommendations

Based off the quantitative data from the analysis and general inferences drawn, the following recommendations are being made

Design

- Identify the unknowns part of the design process. Negotiate the requirement as early as possible, in light of these unknowns, with the stakeholder before continuing

- Prototype/Pilot the unknowns before locking the design. Active developer participation in prototypes to capitalize learning.

- Identify and engage the specialists (internal or external) at the right stage

- Make high & low level design mandatory. Inclusion of parties with right/required level of expertise (HLD-> Subsystem level to be performed by team lead; LLD -> Component level by developer)

- Formal design review and a sign off before the implementation (Tech Lead/Architect)

Recommendations

Based off the quantitative data from the analysis and general inferences drawn, the following recommendations are being made

Implementation

- Formal low level design and review of the same prior to implementation. Low level design should cover enough to depict implementation logic, use of common components/practices, coverage for the features/functions

- Mandatory code review before component is released for QA. Wherever possible, adoption of tools to expand the coverage of the code base

- Define and establish a common understanding of the unit testing coverage prior to implementation

- Formally track the unit testing activities, results. Leverage the results in the readiness of follow on activities

- More meaningful notes when resolving a ticket (Most commonly observed note is "Fixed")

Recommendations

Based off the quantitative data from the analysis and general inferences drawn, the following recommendations are being made

Requirements

- Formal review of requirements Dev/QA, where these can be identified and addressed

Quality Assurance

- QA to validate requirements against the locked down scope, for gaps and lack of clarity
- QA to be involved earlier in the requirement process

Build, Environment & Misc

- Release Manager to analyze the dependencies and assess the readiness of a deployment
- Quick shakeout of the deployment before carrying out the normal business

Q & A Time

Appendix

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Requirements	Lack of clarity in the Requirement	Formal review of requirements Dev/QA, where these can be identified and addressed
	Missed Requirement	Formal review with stakeholders QA to validate requirements against the locked down scope
	Conflicting Requirements	Formal review of requirements Dev/QA, where these can be identified and addressed QA to validate requirements against the locked down scope
	Change in the requirement not reflected	Adoption on change request process for every change past the baseline QA to validate requirements against the locked down scope
	New/Late requirements	Adoption of CCB process within the track as well More emphasis on the upfront analysis of the change before making a decision on the CR

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Design	Inadequate/Poor design	Make high & low level design mandatory. Inclusion of parties with right/required level of expertise Formal design review and a sign off before the implementation
	Non functional requirements are not taken into consideration	Gather and socialize the non functional requirements Factor and validate the coverage for non functional in design
	Design option has lot of unknowns	Identify the unknowns as part of the design process. Negotiate the requirement, in light of these unknowns, with the stakeholder Prototype the unknowns before locking the design Incorporate the specialists (internal or external) at the right step

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Implementation	Gaps in functional understanding	Ramp up sessions on critical components - Ex: Data Models, Framework Models, Overall view of the system, demo of the prototypes as applicable Additional ramp up time for new resources
	Poor/sub-optimal implementation choices	Formal low level design and review of the same prior to implementation. Low level design should cover enough to depict implementation logic, use of common components/practices, coverage for the features/functions Mandatory code review before component is released for QA. Wherever possible, adoption of tools to expand the coverage of the code base Factor in and provide the feedback on the quality levels, delivered by an individual, during project close out, performance reviews and/or at the right time as appropriate
	Incomplete implementation	Exclude from releasing the component to subsequent steps. Assess the readiness of the build itself based on the functionality avaliable If it's vital to continue with an incomplete component, release/build notes should include a clear articulation of what's in and out Project Managers to (re)assess the appropriateness of the scheduled builds (more so if they are aggressive like daily or 2-3 builds a week)
	Omission/Overlooking	Mandatory code review before component is released for QA. Wherever possible, adoption of tools to expand the coverage of the code base

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Implementation	Unit testing is not adequate	Identify, plan the environment, data, external component needs at the planning stages Define and establish a common understanding of the unit testing coverage prior to implementation. Formally track the unit testing activities, results. Leverage the results in the readiness of follow on activities Leverage QA artifacts, resources to improve the rigor. Or tag dev/qa members to test others code
	Implementation doesn't support desired variability	Formal design review and a sign off before the implementation Mandatory code review before component is released for QA. Wherever possible, adoption of tools to expand the coverage of the code base
	Ripple effect of a change	Assess the implications of the CR Define and establish a common understanding of the unit testing coverage prior to implementation.
	Gaps in behavior of the underlying components	Prototype the unknowns before locking the design Incorporate the specialists (internal or external) at the right step

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Testing	Gaps in the understanding of scope of test build	Project Manager and or Dev lead to clearly define and communicate the scope of the build at the planning time as well as at hand off time
	Test bed doesn't comply with expected	Test plan to the details of the test environment, with specific references to the system Assess in light of the insight from design Assess the specific needs as applicable to the component
	Gaps in functional understanding	Ramp up sessions on critical components - Ex: Data Models, Framework Models, Overall view of the system, demo of the prototypes as applicable Active QA participation in requirements review (JAD Sessions)
	Test condition is beyond the reasonable stretch	Review the test cases, prior to the actual QA, with RA and developers to identify and resolve these ahead of time
	Test scenarios are not adequate	Test plan to define the specific bounds of the of testing across various QA tracks (system, integrated, operational….) to identify any gaps or overlaps Test plan review with all appropriate stakeholders same time to get a feel for comprehensive view (business operations, system operations, business users?..)
	Redundancy	Test case reviews - Identify and address redundancy in the test cases QA lead/manager to take lead on analyzing the gaps/overlaps before the actual commencement of QA activities and address them ahead of time Develop a tool/artifact to a comprehensive view of the QA coverage across tracks

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Environment	Dependent components are not in the expected conditions	Make high & low level design mandatory. Inclusion of parties with right/required level of expertise Release Notes to call out the dependencies on external components/builds Release Manager to analyze the dependencies and assess the readiness of a deployment Quick shakeout of the deployment before carrying out the normal business
	Deployment of wrong components	Review of deployment BOM (build of material) Active participation and communications of the liaisons
	Incomplete/Inaccurate deployment	Adherence to the deployment BOM Training for the new members; or new members to be shadowed by seasoned players

Clusters (Common pitfalls)

Category	Issue	Mitigation Strategies
Misc	Build related issue	Clear documentation of build process, review it with CM team and handle the build with a primary designated party (or a planned backup)
	Component releases are not properly aligned	More rigorous program level plan that actively reflects and tracks the build out and rollout dependencies across tracks at the required level of details Integrated build schedule to be managed at the program level