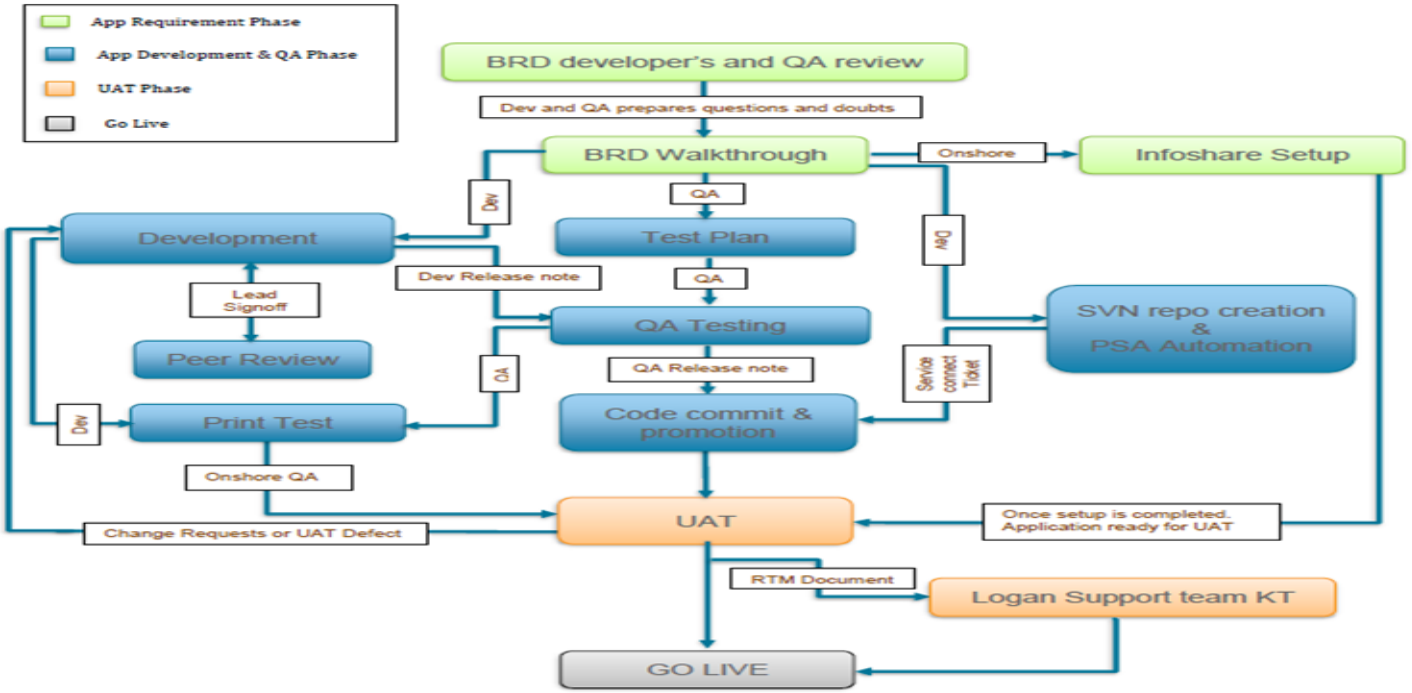
TEKsystems Global Services

Codebase and Process

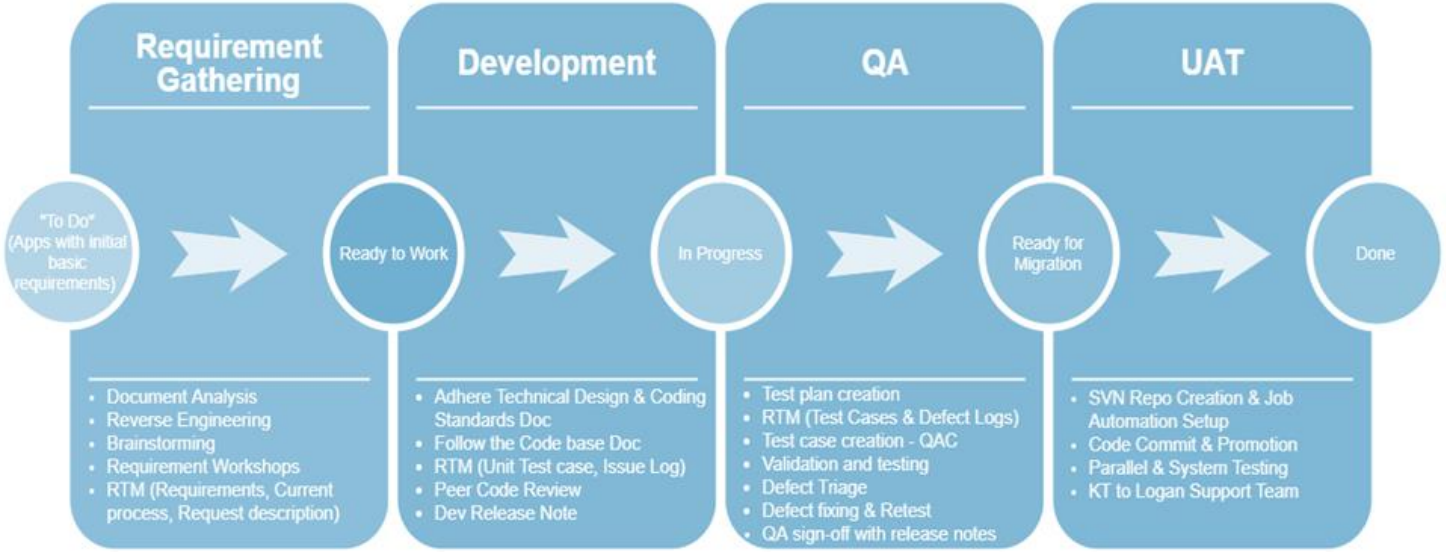
Presented by: Anand JUjare

Date of Presentation: 9/2/20

Sample flow chart



Process overview



Coding standards

- The first important thing to note is that a coding standards section or document is not about right and wrong. It's not about good and bad or which method is better.
- A coding standards document's purpose is to make sure that all code is designed, written and laid out the same to make it easier for a developer to switch from one person’s work to another without the needed change of mentality to read someone else's style.
- We should try to make the code self-documenting.
- The simple rule is to include comments for all of the steps
- The scripts are written for automation also needs to be properly commented and self-descriptive.
- Scripts’ naming convention should be meaningful and self-explanatory.

Coding standards contd...

- Spend time formatting code so it looks visually appealing. Ugly code is hard to read.
- Provided that, we should leave a blank line between sections, indent consistently, and avoid exceeding 90 or so columns of text, control files can be surprisingly readable.
- Also, we should pick meaningful, human-readable names for variables and modules. For example, certifiedMail.reports is a better name than cm.rpts. The specific naming convention is not important -- just come up with something and we should stick to it.

Best practices

Best Practices which needs to be followed	Comments
Improved readability (margins, white space, tabs)	
Code block starting and ending	
Commented code removed	We should not keep the unwanted codes in the control file instead of commenting it we should remove the extra code as well which can be added default by the Appbuilder but not being used.
Naming conventions followed	
Hard coding, constants	We should not hard code any variables in the control file or in script instead we should have declared it in default variable and always use the same variable wherever it is required.
Valid comments (why instead of what)	Always add a descriptive comment to each section of the code
Multiple loops and if-else blocks	Try avoiding multiple loops as it makes the code execution efficiency affected.
Normalization	Only for PDF input files.
Email	Do not code the email ID's in the script. Save it in default variable list and always take it as a parameter to the script.
Reusable functions and services	Using the module instead of customizing it using Perl code. Always use the one module instead with proper assign variables instead of having many multiple modules
Repo cleaned up	Not to keep more than 50 runs and also not to keep backup files or multiple files in not needs directories.
Tests are up to date with the latest commit	
Are all requirements coded	We should always check if all the requirement are coded in the control file or not

Functional practices

In Processes Section

- In “arcrep” variable add at the end - ${runpath}pre_process.snap
  - For example

```
arcrep="30;$runpath$site$lvl.$sbu.$pi$n.$pjt.ptkcat$post.pdf,$
u.$pi$n.$pjt.repext*,${runpath}batinf.txt,$runpath$site$lvl.$sbu.
runpath}*.objects.arr,${runpath}*.pages.arr,${runpath}pre_proc
```

- Chunk quantity should be declared in this section based on the va
  BRD document.

```
chunk_qty=""
```

- Below variables should be set as following.

```
brk_type="documents"
email_error="y"
```

Functional practices and signoff

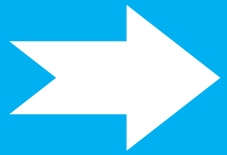
- **Fonts**
- Default Powerstream font: C0HE09R0
- Helvetica font can also be used because it is usually available on all printers.
- Quality/Z records should use the same font that the client uses in the file for addresses.

Name	Signoff Team
BRD Requirement Signoff	Respective application's PM and BA
Developers Signoff	Development Team
QA signoff	Offshore QA
Print Test signoff	Operations Team & Onshore QA
UAT	Clients and Operations Team
Support Team KT signoff	Logan support Team
PM and VAST signoff	Business and clients' PM
Change Request Signoff (Optional)	Respective application's PM and BA

Codebase

- Create a repository of the common modules used in the application control file creation.
- A collection of source code used to build a particular software system, application, or software component.
- Stored in a version control system like Subversion
- Git is a distributed version control system

Requirement Gathering



- Document Analysis
- Reverse Engineering
- Brainstorming
- Requirement Workshops
- RTM (Requirements, Current process, Request description)

- **Document Analysis: Client** provides the initial and basic requirement in this document known as BRD.
- **Reverse Engineering:** As the requirements are not properly captured, Clients provide us the existing codes (which are written using different languages and platforms) to perform reverse engineering for capturing the requirements.
- Brainstorming: Brainstorming is used in requirement gathering to get as many ideas as possible from the leads and other team members for the application development. Generally used to identify possible solutions to problems, and clarify details applications and their integration with 3rd party tools.
- Requirement Workshops: All the selected stakeholders (client SME's, TEK developers etc.) collaborate together in this meeting to discover, refine, prioritize, validate and discuss the requirements.

Requirements traceability

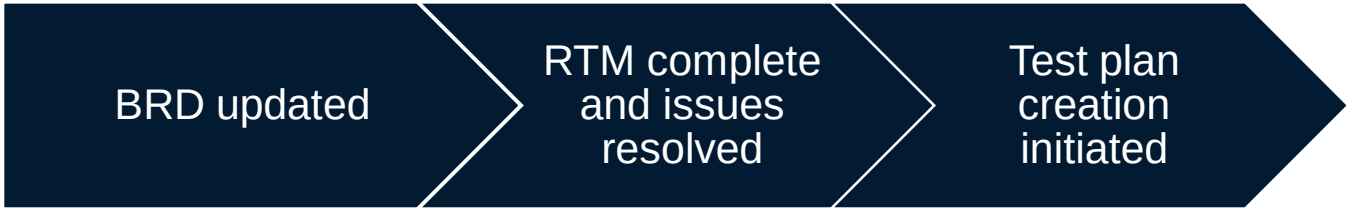
In RTM, we capture all client requirements and their traceability in a single document delivered at the conclusion of the application life-cycle. This document that maps and traces all the user requirement with Request description, Existing Process, Test cases, Issue logs, Defect logs, Unit test cases, Change requests etc.

Requirements need to be elucidated regardless of the delivery model

Once a functional requirement / user story is groomed, it needs t be elaborated

- Normal flow
- Alternate flow
- Exceptions
- Validations
- Wire frames
- Error handling
- Database field mapping
- CRUD operations

Checkpoints



- Reduce technical debt
- Technical debt results when developers fast track delivery of code that will have to be refactored.
- Upto 25% of a sprint is used to refactor technical debt

Development

- Adhere Technical Design
- Adhere Coding Standards Doc
- Follow the Codebase
- RTM (Unit Test case, Issue Log)
- Peer Code Review & Risk Analysis

❑ **Technical Design Document:**

Technical Design Doc

❑ **Code Base Document:**

TEKSystems_Codi ndards&Codebas

CodeReview_Che cklist.xlsx

❑ **Risk Analysis Document:**

App Risk Tracker

❑ **Dev Release note**

Dev Releasenote Doc

QUALITY CHECKPOINT

Below are the checkpoints after development is completed and before moving the application to Quality assurance.

All team leads together makes sure that below checkpoints are met.

- **Code Review:** Application code needs to go through a review process to check that the standards are met and code validated for maintainability, reusability, scalability and efficiency of the application.
- **Risk Analysis:** At the time of code review, leads also review the risks and come upwith a mitigation plan.
- **Dev Release Note:** Dev release note should be filled before developer hands over the application to QA.
- **Test Plan and code promotion:** QA creates a test plan in parallel while development is going on. Before starting the testing QA needs to get the Test plan approved. Documents are created and automation ticket is created.

QA

- Test plan creation
- RTM (Test Cases & Defect Logs)
- Test case creation - QAC
- Validation and testing
- Defect Triage
- Defect fixing & Retest
- QA signoff with release notes

QA team derives and/or select test cases based on an analysis of the specification given in the BRD and the final RTM document, either functional or non-functional, of a component or application without reference to its internal coding structure.

- **QA Tracker Sheet:**
  https://drive.google.com/open?id=1lb7mq63k2JwM1zXuYqIhLGieCAoQEoAFvTUcoo-T2dI

- **Test Plan & QA release note Document:**


Test Plan Doc


QA release note

- **Defect Triage Documents:**


QA Defect Tracker


QA Defect Triage Details


QA defect Triage Process doc

Below are the checkpoints after QA is completed and before moving the application to UAT

- **Defect Triage Meet:** After the application testing is completed. QA holds a triage meeting with leads and developer to discuss the defects raised. So the common defects can be ignored in future.

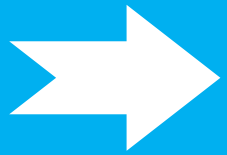
- **Update the defect tracker sheet:** All the defect should be captured in defect tracker sheet and also in QAC.

- **Print test:** Sample printing of out file for QA team review.

- **RTM Filling:**. Developer and QA should capture all the information collected during the development and QA phase.

USER ACCEPTANCE TESTING

UAT



- SVN Repo Creation
- Job Automation PSA Setup
- Code Commit & Code Promotion
- Parallel & System Testing
- KT to Logan Support Team

The PSA group runs a fully-automated test through the automatically triggered run by the scheduling software.

Validate that all productions report and files are generated and printed/transmitted successfully.

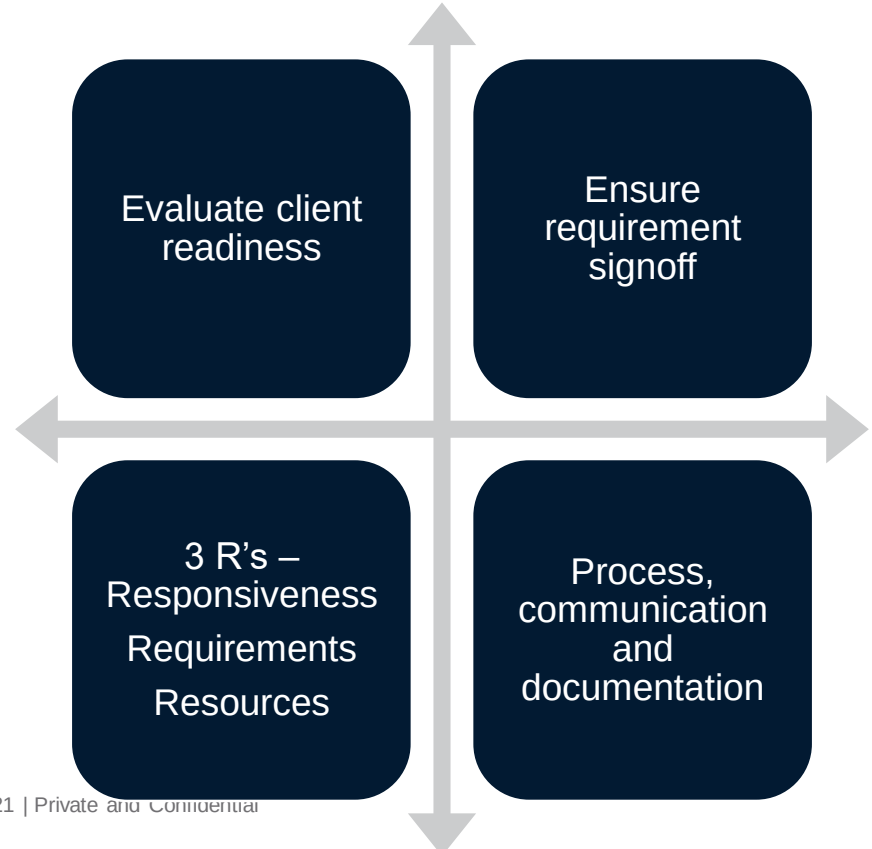
Have production print and insert at least 1100 documents to validate quality records, imaging, inserting, mail sorting and postal attributes.

Knowledge Transition to Support Team:
Total overview of the application will be given the respective programmer to the support team.

Following are in the Scope of KT

- Functional requirement
- Non-functional requirements
- Code walkthrough

Takeaways



Good to know

- **Design patterns**
- **Design thinking**

TEKsystems
Own change

THANK YOU