SDLC Processes in micro-services based architectural approach

A compilation of our Toolchain, Standards & Processes
Business domain overview

• Non-communicable diseases are the leading cause of insurance claims*

• Differential insurance model.
  Pay less premium if you follow healthy life-style habits like actively exercise, eat healthy food, don’t smoke or moderately drink alcohol.

*https://en.wikipedia.org/wiki/Non-communicable_disease
Product domain

Healthy food bills
Fitness devices
Health Assessments
Participation in events like marathon races

Kind of data that we collect

micro-services currently in production

activity purchase agreement/policy
product finance party/member
assessment reward notification
goal points cashback
Our Team

• 14 member team consisting of:

1 project manager
3 Business Analysts
1 Scrum master
9 developers [South Africa (7), India (myself), Belarus(1)]
SDLC process overview

What we inculcate as part of our SDLC process:

• Respect (for each other’s capabilities)
• Integrity (agile)
• Empathy (for end-users)
• Excel and continued thirst for knowledge and learning new tools
• 10 principles of leadership*

(Strict) definition of done!

<table>
<thead>
<tr>
<th>Level</th>
<th>DOR</th>
<th>DOD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint</td>
<td>Clear sprint goal(s)</td>
<td>All Product Backlog Items, adhering to the DoD for the specific team, have been deployed to QA</td>
</tr>
<tr>
<td></td>
<td>Prioritized Backlog</td>
<td>All automated tests have passed in QA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All defects logged during functional testing have been fixed. Unless it was deferred</td>
</tr>
</tbody>
</table>

**Analysis:**
- Use Case must be complete and peer reviewed. It should include, where applicable:
  - A Fable Spec
  - Definition of Interface
  - Activity Diagram
- Domain Model must be complete and reviewed by both Greg Gilbert and Byron
- Test Scenarios have been identified and documented in QC
- New user stories have been created or existing user stories have been updated with acceptance criteria

**Design:**
- The following artifacts have been created and reviewed by the applicable people:
  - Rules design – peer reviewed
  - ERD – peer reviewed
  - Component dependency diagram – reviewed by Greg Gilbert and / or Wendy Billay
  - Braeger Interface – peer reviewed & reviewed by Greg Gilbert and / or Wendy Billay
  - Activity Diagram (if needed) – reviewed by Greg Gilbert

**Development:**
- All the acceptance criteria for a user story have been met
- Sonar Technical Debt issues have been fixed, before the code is pushed for peer review
- Code have been peer reviewed by a member of the team
- Dev testing has passed on the Dev Environment and the results are documented and saved in Confluence. Dev testing should include:
  - Unit Tests that provide 80% test coverage
  - Automated Regression Tests
- Release notes and deployment instructions have been created / updated and saved in Confluence
- If applicable, the following should have been done:
  - Physical ERD generated
  - Rules Implementation
  - Domain changes
  - Configuration & Rules setup – available in the repo
  - DB changes??

**Testing**
- Detailed Test Cases, corresponding with a story’s acceptance criteria, have been created in QC
- Test run have been identified
- Test results have been updated and reviewed in QC
Documentation

- Requirement specifications links
- Checklist (code review etc)
- Standards (Logging, REST APIs etc)
- Testing (integrated, regression)
- Deployment instructions
- Release notes and versioning
- Infrastructure-as-code
- Process documentation
- Standby resource POCs etc
Requirement Specification repository in Enterprise Architect
Model driven architecture. Model is exported from EA tool into Java code by Business Analysts.

Since we use functional Java 8, Spring-Hibernate acts as a glue-code for Service/DTO/DAO layers while all business process logic is into “process”
Review Standards

Code Review Checklist

- Overview
- Meaningful Names
- Functions
- Comments
- Formatting
- Objects and Data Structures
- Error Handling
- Unit Testing
- Classes
- Effective Data
- General
- POM Files
- Faign
- SQL
- Swagger Interfaces

Overview

The following checklist is compiled taking "Clean Code: A Handbook of Agile Software Craftsmanship by Robert C. Martin" as input. The list below has been adjusted to DPM standards. For more information/examples consult the Clean code book if something isn't clear. Items in orange are items that we are more lenient on.

Meaningful Names

- Use Intention-Revealing Names.
  - Names shouldn't require comments. If it does, the name doesn't reveal its intent.
  - Avoid Disinformation

  ```java
  //non compliant - this isn't actually a list
  Map<Integer, Account> accountList;
  ```

- Use Meaningful Distinctions
  - Don't append number series or noise words to appease the compiler.
Integration testing

• The best part of Spring4JUnit is Integration Testing !!
Commit messages with JIRA and testing document link
Continuous Integration

- Jenkins Build on Push and SonarCube
Automated testing

- Swagger+Groovy
- Currently, we are trying to figure out how to test when there is ESB (tibco) Queue in between.
We want more essential documentation
Currently, we are figuring out how client feedback can propagate back to changing the requirements specification in EA tool.
DevOps

https://fabric8.io

- Moved to RedHat Openshift v3.3 Container platform on AWS IaaS recently from v2
- Progressing towards Infrastructure-as-code
- We use bash scripts and DSL (domain specific language) like Go for Jenkins plugins
• Jenkins-CI pipeline in action!
Our challenges where we need improvement

• Systemic thinking* vs silo thinking
• People re-organisation around business capabilities instead of older multi-tier architecture layers
• Tracing and effective monitoring infrastructure
• Essential Logging (currently ELK stack is not able to cope with the humungous logs)
• End-to-end testing (API-Gateway to API-Gateway) despite business rules engine and ESB queue in between :(

Thank you

• Sandesh Gandhi
  Build apps that users would love to use

sandugandhi@gmail.com

https://radicalvision.wordpress.com