Integration Development Manual

Coop One

**Version History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Updated by | Revision | Purpose |
|  |  |  |  |
| 18/04/2018 | Rasmus Kjelsmark Olsson | 0.1 | Initial draft |
| 19/04/2018 | Lokesh Muppana | 0.2 | Updated Working model |
|  |  |  |  |

Table of Contents

[1. Introduction 3](#_Toc512850598)

[2. Coop One Program 3](#_Toc512850599)

[3. Integration Development Team organization (Anders, Rasmus) 4](#_Toc512850600)

[Overview 4](#_Toc512850601)

[Documentation and gates 4](#_Toc512850602)

[Planning 5](#_Toc512850603)

[KPS planning as input to integration planning 5](#_Toc512850604)

[Wave (release) planning 6](#_Toc512850605)

[Execution 8](#_Toc512850606)

[3.2.2 Feature/Story planning and development 8](#_Toc512850607)

[Source Control, environments and branching 10](#_Toc512850608)

[Operation 10](#_Toc512850609)

[Ramp up 10](#_Toc512850610)

[2. Working Model (Lokesh, Anders) 10](#_Toc512850611)

[3. Functional and Technical design (Lokesh, Anders) 11](#_Toc512850612)

[4. Programming Guidelines (Anders) 11](#_Toc512850613)

[5. Tools (Anders) 12](#_Toc512850614)

[6. Operations (TBD) 13](#_Toc512850615)

# Introduction

This document covers the COEX & To-Be CLA architecture in Coop. CLA supports Coop Category in the creation of standard supplier agreements, which covers procurement of goods and standardized services. The agreement includes also standardized services that Coop may provide to suppliers e.g. access to sales data, space management in Coop stores etc. for which CLA also creates the invoice for the services rendered.

# Coop One Program

# Integration Development Team organization (Anders, Rasmus)

## Overview

|  |  |
| --- | --- |
| **Step** | **Documentation level** |

## Documentation and gates

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Deliverables** | **Focus on** | **Gate** |
| Kick Off - 2 |  | Finding 3.party integrations |  |
| Integration design - 3 | 1 document pr. Feature in sharepoint incl. first draft of end2end mapping. | Finding integrations points | Overall design review  (pr. Epic) |
| Release Planning - 4 | 1 document pr story in sharepoint. | Finding gap’s in mapping – etc. missing fields for mapping into legacy | Overall Integration review  (pr. Feature) |
| Iteration Planning - 5 | Extend document in share point | Error handling and stability | Integration review  (pr. story) |

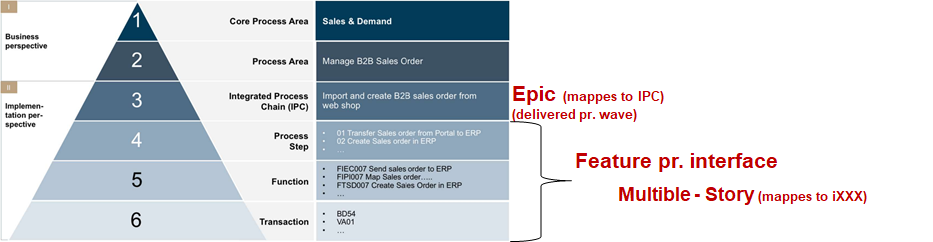
## 

## Planning

KPS planning as input to integration planning

The Integration planning of task, will have the following structure:

* Epic
  + Feature
    - Story
      * Task
    - Bug



1. We will plan to have 1 epic pr. IPC (that haves interfaces related to it)
2. We will plan 1 feature pr interface (iXXX)
   1. But because an iXXX can be part of multible IPC's the iXXX will be registered at the story level

Example:

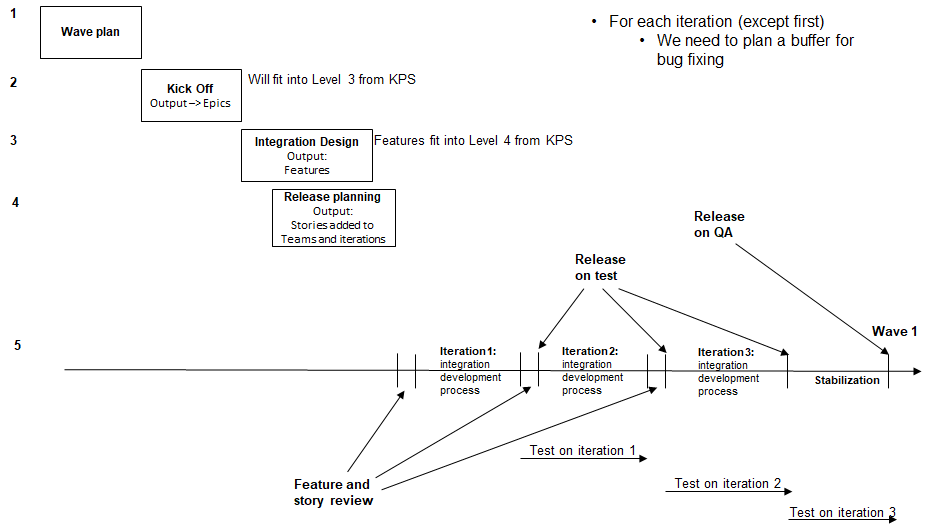


### Wave (release) planning

KPS has divided the delivery into waves. Integration will consider these waves as target for deliveries, in common word -> releases.

Example:

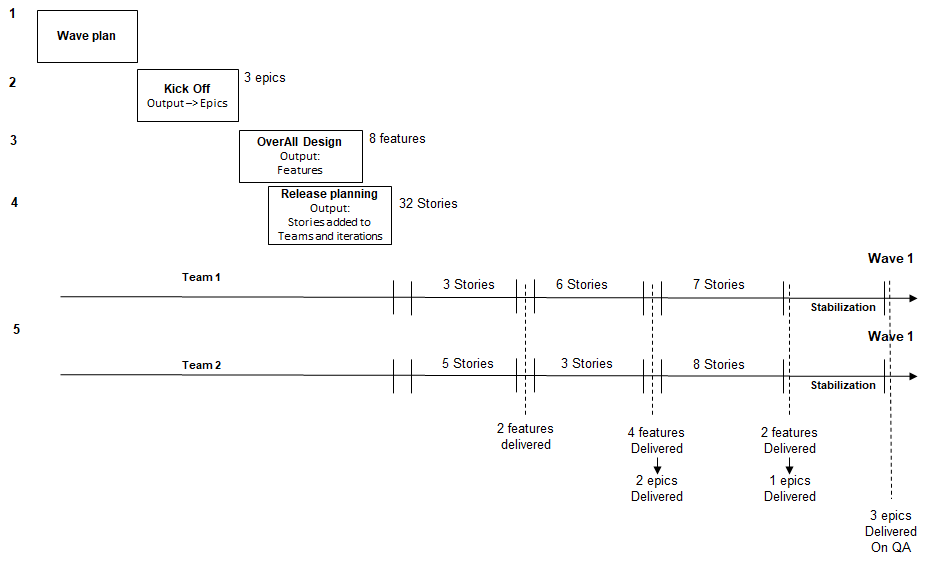
Wave 1 will be considered as release 1, so we will plan our work to be delivered at the start of each wave (and not the end of each wave)



In the following table focus is on integrations, but 3.party, migration etc. must have similar steps/outputs for 2-5.

|  |  |  |  |
| --- | --- | --- | --- |
| Step | input | output | actors |
| 1. Cluster plan |  | IPC's mapped to each wave | KPS |
| 1. Kick Off | IPC's | For integration:   1. Overall design in Qualiware 2. Epics created and described in VSTS | KPS, Integration(TCS/Coop), migration, 3.party |
| 1. Integration Design | Epics | 1. Overall design of each integration (feature) 2. Feature created in VSTS with rough estimate 3. Documentation created in Sharepoint and linked in VSTS and qualiware 4. End2end mapping document | Integration(TCS/Coop), KPS, 3.party |
| 1. Release planning | Features | 1. Detailed design of stories (could also include first Mapping draft) 2. Stories created in VSTS with rough estimate 3. Stories are planned in iteration's | Integration(TCS/Coop) |
| 1. Iterations | Stories | 1. For each iteration    1. Task created with estimates in VSTS   Explained in details in the 'Execution' section | Integration(TCS/Coop) |

## Execution

VSTS Structure

#### Team Structure

* System Integration
  + Team 1
  + Team 2
  + …

#### Item Structure

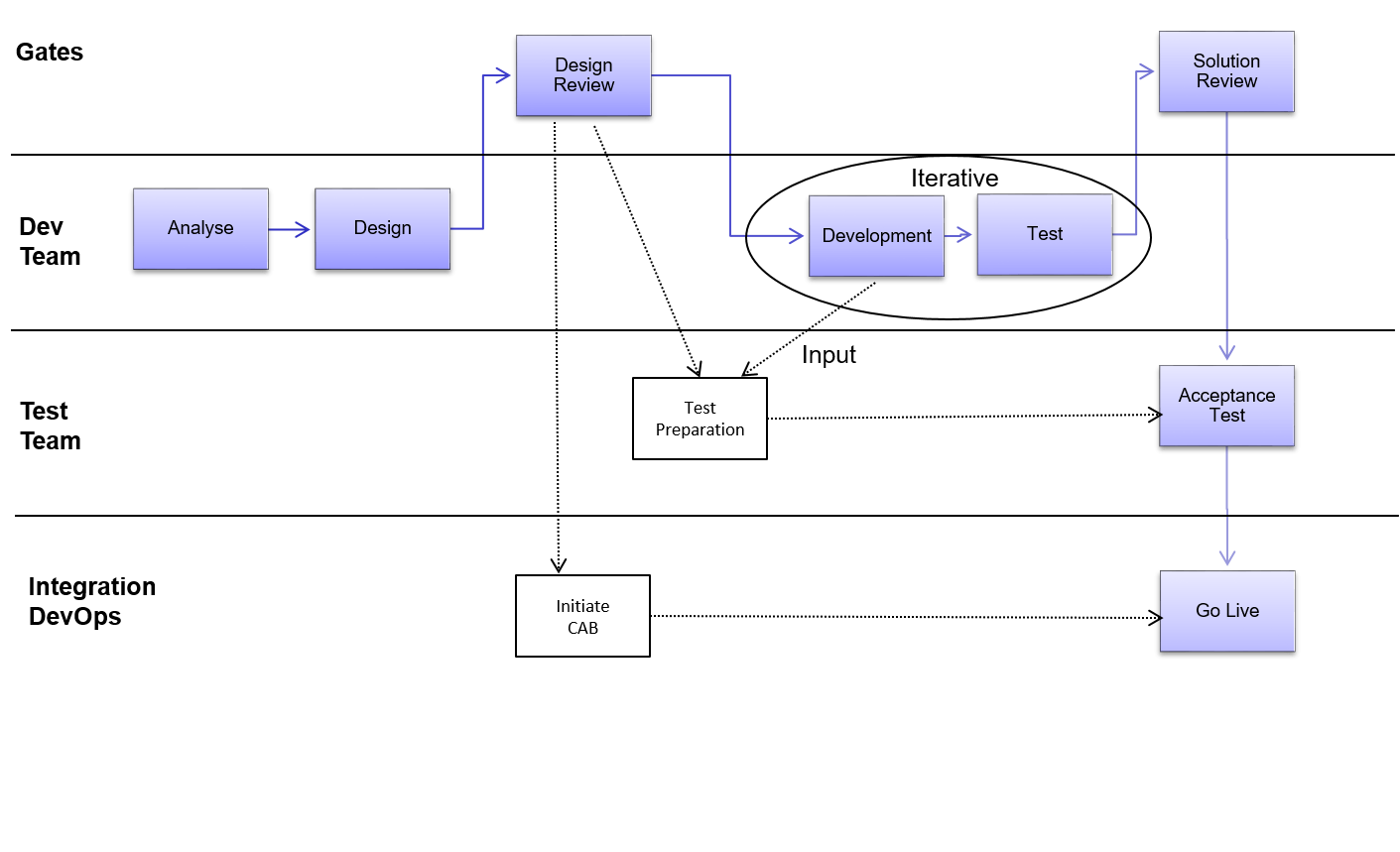
An Item is one of: Epic, Feature, Story, Bug, Task.

The table shows where each item is created

|  |  |
| --- | --- |
| System Integration | Team X |
| Epic  Feature (on an Epic)  Iteration (same for all teams, having stories) | Story (on a feature)  Task (on a story)  Bug (on a feature= |

All Epic/feature/iteration in ‘System Integration is inherited by all the underlying teams.

### 3.2.2 Feature/Story planning and development



Source Control, environments and branching

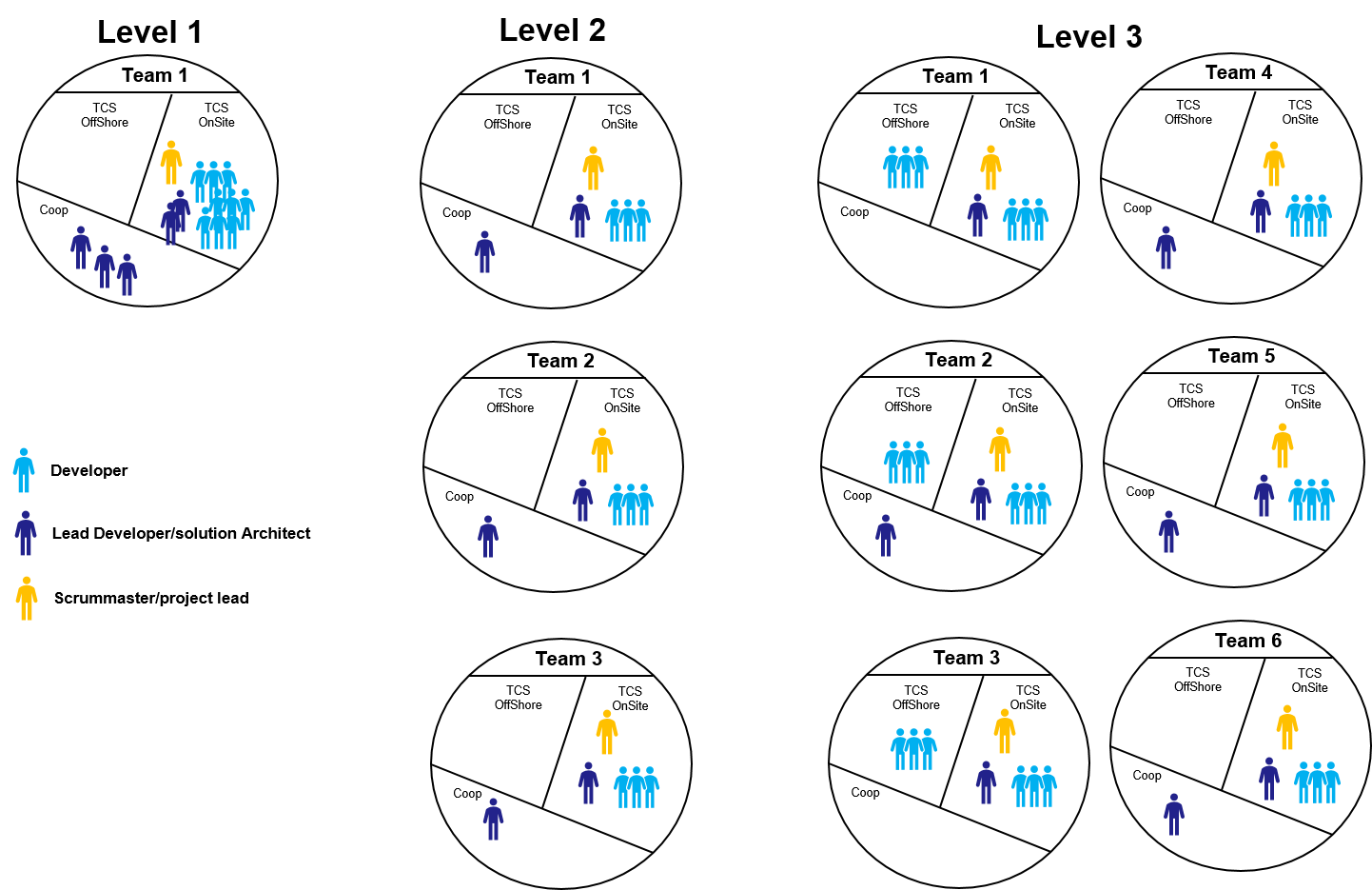
We will use GIT flow as our branching methodology.

See <https://danielkummer.github.io/git-flow-cheatsheet/> to understand this method.

There will be 4 environment in System Integration: DEV, TEST, QA and PROD

## Operation

## Ramp up



Timeline/proces – Design > hand-over > delivery > … Exit criteria

# Working Model (Lokesh, Anders)

|  |  |
| --- | --- |
| Task | Status |
| Assignment of Waves to IPC | Done |
| Assignment of IPC to RICEF | In Progress |
| Assignment of RICEF to Interface ID | As part of the waves |

Note: RICEF ID is different from Interface ID. Both must have different naming conventions. What we have now are RICEF IDs. Interface ID can be assigned during the wave planning and must be documented in Qualiware(or it’s alternative) – Who owns it?

|  |  |
| --- | --- |
| Solution Manager | VSTS |
| IPC | Epic |
| RICEF | Feature |
| Work packages | Stories |

* Above example is for the following case
  + One IPC with multiple RICEF objects
  + Some of these RICEF objects belong to same Interface ID(IF001)
* Stories will be assigned to teams who will go through the full iteration cycle
* Work items(feature/story) in VSTS can have application name as a tag
* We can create stories under each feature for legacy application changes
* KPS can track status of Integration development by checking status of the epics
* One interface can be part of multiple Epics(IPC)

|  |  |  |
| --- | --- | --- |
| Epic(IPC) | Poslog Interface | Wave |
| Retail Sale | Yes | 2 |
| Goods movement | Yes | 3 |
| Stock adjustment | Yes | 4 |

Integration List (with Example) incl. reporting - – EXCEL LIST

Standards (on Integration) –

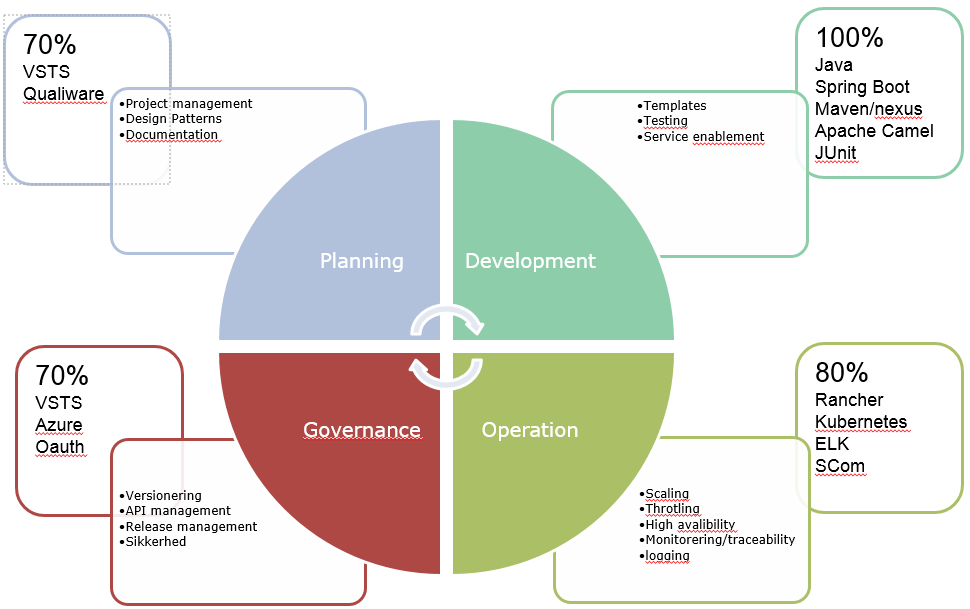
Defect proces

# Functional and Technical design (Lokesh, Anders)

# Programming Guidelines (Anders)

Rules, design patterns

# Tools (Anders)



|  |  |  |
| --- | --- | --- |
| Capability | Area | Tools |
| Planning |  |  |
|  | Process management | VSTS |
|  | Overall design of epics | Qualiware |
|  | Design documentation (feature+story) | Sharepoint |
| Execution |  |  |
|  | Development platform | IntelliJ or Eclipse |
|  | Source control | VSTS (GIT) |
|  | Key Java API | Maven, Spring, Camel |
|  | Storage technologies | Rapid MQ, MS-SQL |
|  | Test | JUnit, Mockito, Spring-test |
| Operation |  |  |
|  | Logging and tracking | Kibana |
|  | Release management | VSTS |
|  | Operation environment | Docker, kubernetes, Rancher |
|  |  |  |
| Governance |  |  |
|  | API management | Azure API mgr. |
|  | Security | OAuth |

Project site

Access

Release

Code control

Offshore Connectivity

Developers from offshore need access to remote desktops with below list of softwares installed in it.

* Java 1.8
* Maven
* Git bash
* Intell IJ
* Rabbit MQ

Remote desktops do not need access to development/test servers. Developed code will be pushed to Git and CI/CD pipeline will deploy the code to dev, test servers. However, remote desktops must be having access to the following

|  |  |  |
| --- | --- | --- |
| Access | Purpose | URL |
| Coop artifact repositories | To have access for Coop central repositories. This is required for maven build | <http://artifact-repository.intra.coop/content/repositories/coop-releases/>  <http://artifact-repository.intra.coop/content/repositories/coop-snapshots/> |

I am assuming that remote desktops will have Internet connectivity to access sharepoint and VSTS/GIT.

|  |  |  |
| --- | --- | --- |
| Sharepoint | To view and edit project documents | <https://azurecoop.sharepoint.com/sites/pwa/CoreERPManagement/default.aspx> |
| VSTS/GIT repositories | To manage code | <https://coopitdevelopment.visualstudio.com/Systems%20integration/> |

Remote desktops must preferably have similar hardware configurations as developer PCs issued to developers at onsite

# Operations (TBD)

Transactions,

Governance tracking