Revenue and Efficency

ideas for SGW

# Continuous delivery:

* 1. Current SGW SDLC model and shortcomings:

A large part of the planned effort today is spent on some redundant and/or non-optimized activities for each release. Will try to quantify this effort compared to the overall avrage effort for all projects.

* Today QA are performing a manual validation of new features, and this is time consuming (About 60 – 70 % of overall QA validation effort)
* Redundant partial NRT phases are planned for each release. Plus, it is usually a partial NRT only that is planned for each cycle, where QA will spend manual effort testing same scenarios for each release during this NRT.
* Release docs are manually updated for each release, which is time consuming for all members (BA, QA, and Dev)
  1. Impact of the current SDLC:
* Customer are unhappy with the current time to market performance; where it takes too long to update a SGW for a market
* Time to fix is also slow because of the planning overheads, i.e. Redundant and non-optimized activities for each release
* Legacy releases like C21 and OCC for CV are not getting updated for years due to the cost of ownership (Too much effort between planning and implementing changes to legacy code, due to lack of modularity and non-well updated specification documentation case of OCC and C21) 🡪 Move to CV adapter would reduce this burden
  1. Proposed solutions:

Adopt a Continuous delivery approach for all SGWs. This means that any feature that is developed & validated will be ready for delivery e.g. by end of Sprint or even right away. Today this is not possible due to the large number of manual steps involved in our SDLC as highlighted below:

* QA is validating new features manually, which is time consuming. Today this manual activity can only start when dev has finished the development of the new feature. This usually leads to QA resource contention by end of Sprint and could result in pushing validation to next Sprint(s), leading to risk of delaying deliveries or increasing follow up releases (i.e. SP releases). In addition, if QA finds an issue with the new feature, they will need to wait again for a fix and then redo the same manual testing effort over and over until it is successfully validated, which results in additional increase of the lead time for customers and for releases.
* In the other hand, NRT validation is performed manually and results in further delaying the delivery until a successful NRT validation is reached. This is also hindering continuous delivery
* Release documents are updated manually, and leads to additional lead time (LT) to overall delivery which is also time consuming and blocks the continuous delivery as well

We propose to optimize ~ 40 – 50% of the overall effort (currently ???% of total effort)

How - To reduce this effort, we propose following initiatives:

* + 1. Automate new feature validation:

Instead of manually validating new features, QA will focus on writing the automated test scripts for the same feature(s), while Dev is still implementing it. To write and run the tests, QA team will use an automation tool that is lively connected to the exchange. A CCP connected automation tool initially branded as SGW Live Testing Platform (SLTP), will allow the validation of the automated scripts and present the following advantages:

* It will remove the sequential dependency between QA and Dev efforts for new Dev items. These automated scripts could be launched anytime when the feature(s) is/are ready for validation.
* Given that the tool, i.e. SLTP, will be connected to CCP, there will be no need to collect WF messages, reducing the overhead of managing off line samples, which is very time consuming.
* Also, during the business case study of the SLTP project, we realize that the average effort to write an automated test script for a new feature is much less than the effort required to validate the same feature manually; and could reduce the validation LT by up to 60%.
  + 1. Automate NRT validation:

The same automation tool, i.e. SLTP, could be leveraged to run NRT as coverage gets improved over time. This will spare the QA team extensive manual effort and will incrementally save part of the NRT effort for each release until we ultimately zero the NRT validation effort as coverage gets improved from one release to the next.

* + 1. Automate the generation of the release documentation:

Updating documents to reflect the new scope of a release will be done automatically by leveraging Jira content and TestRail-to-Jira integration. So, instead of updating docs manually, we write content once in Jira and reuse it to automate the generation of the updated documents.

* + 1. Better CI integration

We have already started improving our CI integration about a year and half ago e.g. to include Docker containerization in line with the FIS CD deployment model, and we still need to improve our CI maturity level to:

* Trigger automated jobs where a new binary can be readily packaged with the new release documentation. And get them automatically pushed to the release repository. Currently, we still need to prepare and upload documents manually to Sharepoint.
* Trigger the SLTP automation testing from CI when deploying the SGW to Docker sandbox. Currently, we can deploy Docker containers, but we are validating manually so leveraging CI with automation testing will be a shift to our delivery model

This way, we can extensively reduce our SDLC lead times, while moving towards a delivery model that is more suitable with the FIS CD Continuous Delivery model.

* 1. How to implement:
  2. Target result and how to measure:
* Value saving chain:
  + Value saving on new feature manual validation by using SLTP 🡪 (~ 73%=(15MD-4MD)/15MD). e.g. Moving from a Sprint of 60 MD worth of QA effort to a 60MD \* (1-73%) = 16 MD after automation
  + Value saving in NRT 🡪 100% saving after few releases (e.g. 4(Assuming a delivery of 1 Release/quarter for a market)\*3(Years) = 12 releases). e.g. A partial NRT average manual effort for a release could be zeroed after about three years
  + Value saving for docs 🡪 60 to 70 % of value saving assuming that only 1/3 of the document generation could be automated, including TRN, market config, Clearing doc, QA report. e.g. 5 MD (Dev + BA + QA) per release could be reduced to 1MD per release

So, the total value saving per Sprint after automation 🡪 From 66MD per Sprint (Explanantion: ~ 60 MD (for new Feature testing) + ~ 5 MD (NRT manual Avg effort per Sprint – Assuming an avg 5 Sprints release duration, i.e. 25MD/5 Sprints = 5 MD per Sprint) + 1 MD (Avg effort per Sprint to update docs, i.e. 5MD/5Sprints = 1MD per release in average) to ~ 16 MD (i.e. 16 (New feature validation) + 0 (NRT after ~ 3 years) + 0.2 (for docs)). This is about 50 MD ~ 75% saving on QA effort for each Sprint.

* Be able to improve time to market and boost customer satisfaction, while being in line with FIS CD release strategy (e.g. Quarterly basis releases for each market), and with the corporate vulnerability and security remediation requirements, etc
* Be able to improve Time to fix and boost customer satisfaction
  1. Budget requirement:
     1. Badget to accelerate development for SLTP: 1 Dev (To fast track SLTP development) + 1 Dev-QA (To fast track automation coverage for QA) = 3 additional Team Members
     2. Badget to improve CI integration and Dev Ops for SGW: 1 Dev-QA resource = 1 additional Team Member

**Total** = 3 additional HR resources required for ~ 2-3 years

# Cloud ready SGW

* 1. Current challenges to be cloud ready:
     1. Cost of ownership

This includes cost to run Configuration and Deployments operations on the cloud, where:

* Configuration and Deployment should be streamlined for each customer and for each environment (e.g. UAT, Staging, Production, etc), and no manual work should be required to re-deploy SGWs
* It should be very easy to amend configuration and deployment details, e.g. through use of portal like UI
* Configuration should not be file system based anymore

It also includes cost to run monitoring and control operations of hosted SGWs, where:

* Today, the SGW managed services team would need to be reactive rather than proactive to deal with issues as they happen instead of being equipped with the right remote diagnostics, monitoring, and control tools to get alerted and also to take preventive actions to avoid impact to hosted services and clients.
  + 1. Performance and Availability
* Performance: FIS CD customers are challenging us to improve performance bandwidth, and SGW may become a performance bottleneck as it is not currently designed to scale up.
* Availability: No failover mechanism is currently implemented for SGW
  + If SGW shuts down, we can only recover manually by restarting the SGW
  + If the hosting VM / machine is down, the service will be interrupted for the customer
    1. Inline with FIS CD deployment model
* SGW should not use File system-based configurations and logging as this will require mounting external FS and complicates deployment model. This is not compliant with FIS CD deployment model as per our last discussion with the team last year, where we agreed to change this
* Be able to deploy SGW inside a Docker container 🡪 This is already done