UNIVERSIDAD INTERNACIONAL DE LAS AMÉRICAS

CARRERA DE INGENIERÍA INDUSTRIAL

TRABAJO FINAL DE GRADUACIÓN

Para optar por el grado de Licenciatura en Ingeniería Industrial

Guía metodológica para la Gestión de Proyectos de Desarrollo de Software en el área de Integration General Business Machine (GBM)

AUTORA

Cinthya Badilla Vargas

TUTOR

Ing. Alejandro Leiva González MBA

LECTOR

(Por definir)

SAN JOSÉ, COSTA RICA, ABRIL 2018

CONTENIDO

Índice de figuras 14

Índice de tablas 17

Dedicatoria 19

Agradecimientos 20

Solicitud de defensa 21

Carta de autorización del director de carrera ..... **¡Error! Marcador no definido.**

Hoja de aprobación del Tribunal examinador ..... **¡Error! Marcador no definido.**

Carta autorización del tutor 22

Carta de revisión filológica 24

Declaración jurada 25

Código de ética 26

Resumen ejecutivo 27

CAPÍTULO I. INTRODUCCIÓN 28

Generalidades de la empresa 30

Visión 32

Misión 32

Valores 32

Planteamiento del problema 33

Objetivo general 33

Objetivos específicos 33

Justificación 34

Antecedentes 34

Proyecciones 35

CAPÍTULO II. MARCO TEÓRICO....36
Herramientas de análisis situación actual....36
Ingeniería de métodos....36
Flujograma....37
IBM BlueWorksLive. 37
Diagrama Ishikawa. 38
Diagrama de Gantt. 39
Tabla multivoto....40
Matriz de Evaluación del Factor Interno (EFI). 41
Estadística....42
Variable. 42
Tipos de variables. 42
Cuantitativas. 42
Cualitativas. 43
Simples. 43
Complejas. 43
Independientes. 43
Dependientes. 43
Intervinientes. 43
Extrañas. 44
Técnica e instrumento....44
Población. 44
Muestra. 44
Metodología tradicional: Método cascada o *waterfall* 45

Metodologías ágiles. 46
Manifiesto ágil. 47
Principios del manifiesto ágil 48
Scrum. 48
Historia. 49
Características de Scrum: Eventos, roles, herramientas, reglas. 49
Eventos. 51
Sprint. 51
Planeación del Sprint (Sprint Planning). 51
Parte 1. 51
Parte 2. 52
Daily Scrum Meeting. 52
Revisión del Sprint (Sprint Review). 53
Retrospectiva. 54
Roles. 54
Product owner. 54
Scrum máster. 55
Equipo de Desarrollo (Development team). 56
Tamaño del equipo de desarrollo 57
Otros roles. 57
Plantillas o artefactos 57
Lista de producto (product backlog). 57
Lista de pendientes del sprint (Sprint backlog). 58
Incremento. 59

Kanbam. 59
Scrumban. 60
Límites del trabajo en proceso (WIP Limits). 61
Tarjetas Kanban. 61
Eventos Scrumban. 62
Sesión diaria (DSU). 62
Sesión semanal (WTB). 62
Roles. 63
Extreme programming. 63
Planificación. 63
Análisis. 64
Diseño y codificación. 64
Testing. 64
Deployment. 65
Roles XP. 65
Design Thinking. 66
Herramientas colaborativas 67
Gestión de proyectos 67
Jira. 67
Team Foundation Server. 67
Gestión de la calidad 67
Practitest. 67
Zephir. 68
Release management 68

Bitbucket. 68
Github 68
Blue Mix con Github. 68
Ingeniería de requerimientos 69
Historias de usuario 69
Buenas prácticas de *software* 70
Peer review. 70
Test driven development TDD. 70
Unit testing. 70
CAPÍTULO III. MARCO METODOLÓGICO 72
Enfoque 72
Diseño (alcance) 74
Investigación descriptiva 74
Investigación explicativa 74
Muestra de la investigación 74
Variables 75
Metodologías 76
Herramientas 76
Instrumentos 79
Proceso para la recolección de datos 80
Método de análisis 80
Cronograma 82
Work Breakdown Structure (WBS) 82
Diagrama de Gantt 83

CAPÍTULO IV. ANÁLISIS DE LA SITUACIÓN 85
Descripción de los procesos actuales 85
Diagrama de proceso Integration 86
Conversatorio del equipo Integration 86
Equipo Integration Core (IC) 87
Diagrama de flujo del equipo Integration Core. 89
Características actuales del proceso. 91
Descripción del proceso de recepción de solicitudes. 91
Proceso de planificación de la producción. 92
Equipo Business Automation 92
Diagrama de flujo del equipo Business Automation (BA). 94
Características del proceso actual. 96
Equipo Software Solutions (SO) 97
Diagrama de Flujo Equipo SO. 98
Proceso actual. 100
Descripción de proceso de recepción de solicitudes. 100
Proceso de planificación de la producción. 100
Diagnóstico madurez actual en prácticas de desarrollo de *software* 100
Levantamiento de requerimientos. 101
Release management. 102
Aseguramiento y control de la calidad. 103
Equipo Integration Core (IC) 104
Práctica de levantamiento de requerimientos. 104
Práctica Release management. 106

Práctica de aseguramiento y control de calidad. 108
Equipo Business Automation (BA). 111
Práctica de levantamiento de requerimientos. 111
Práctica Release management. 113
Práctica de aseguramiento y control de calidad. 114
Equipo Software Solutions (SO). 117
Práctica de levantamiento de requerimientos. 117
Práctica Release management. 120
Práctica de aseguramiento y control de calidad. 122
Diagrama Ishikawa (causa y efecto) 124
Recursos humanos (RRHH). 126
Herramientas. 127
Plantillas. 127
Métricas. 128
Métodos. 128
Comunicación. 129
Análisis de fortalezas y debilidades 129
Tabla multivoto. 129
Matriz EFI. 136
CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES 139
CAPÍTULO VI PROPUESTA 141
Propuesta metodológica 143
Scrumban 143
Propuesta de estructura 145

Modelo de priorización..... 147
Tablero Kanban-volcán..... 147
Propuesta de procesos de desarrollo de requerimientos de *software*..... 150
Guía de estados..... 150
Proceso de solicitudes nuevas..... 150
Solicitud de la necesidad..... 150
Levantamiento de la necesidad..... 152
Planning..... 153
Proceso atención de tiquetes (*Fast Line*)..... 154
Desarrollo, pruebas y entrega..... 156
Propuesta de procesos de prácticas ágiles..... 157
Proceso ágil de gestión de necesidades..... 157
Educción y análisis- Descubrimiento de la necesidad (Inception)..... 157
Alinear y definir expectativas..... 157
Definir la visión..... 157
Técnicas propuestas..... 157
Tablero de visión de producto..... 157
Delimitar y priorizar..... 157
Técnicas propuestas..... 158
El vecindario..... 158
La lista del NO..... 158
Mapa de historias de usuario..... 158
Mínimo producto viable..... 158
Plan de Salidas a producción (releases)..... 159

Elaborar historias de usuario..... 159
Entendimiento de la necesidad..... 159
Construcción base de la necesidad:..... 159
Detalle de la historia de usuario:..... 159
Validación técnica y creación o ajuste del diseño de arquitectura:..... 159
Construcción de la matriz de dependencias:..... 159
Especificar los criterios de aceptación. 160
Refinamiento..... 160
Control y seguimiento 161
Daily Scrum Meeting.. 162
Sprint Review (Revisión de la iteración). 163
Sprint retrospective. 165
Proceso ágil de gestión de la calidad..... 167
Proceso ágil de *Release management*..... 171
Arquitecturas de almacenamiento..... 171
Gestión de versiones. 171
Clasificación de versiones. 171
Elementos básicos de versionamiento. 172
Línea base. 172
Branch (Rama). 172
Merge. 173
Checkout. 173
Commit. 173
Subir cambio (Check in, push)..... 173

Update (Fetch)..... 173
Historial de cambios. 173
Conflictos. 173
Estructura de versionamiento. 176
Nomenclatura de los Branch. 177
Políticas de *Release management.* 179
Manejo de carpetas. 179
Políticas de creación y manejo de branch. 180
Proceso de garantía del código. 183
Propuesta de recursos 185
Propuesta de roles 185
Equipo solucionador. 185
Definición. 185
Competencias y habilidades generales. 186
Competencias y habilidades específicas del perfil de desarrollo. 187
Competencias y habilidades específicas del perfil de calidad 188
Scrum máster. 189
Definición. 189
Competencias y habilidades. 190
Product owner. 191
Definición. 191
Competencias y habilidades. 191
Ingeniero de calidad. 192
Definición. 192

Competencias y habilidades 192
Analista de solución. 194
Definición. 194
Competencias y habilidades. 194
Líder técnico. 196
Definición. 196
Competencias y habilidades. 196
Release manager. 198
Definición. 198
Competencias y habilidades 198
Integration Management Team (IMT). 198
Definición. 198
Competencias y habilidades. 199
Operation Management Team (OMT). 199
Definición. 199
Competencias y habilidades. 200
Artefactos 202
Inventario de necesidades o *product backlog.* 202
Inventario de necesidades del *sprint* o *sprint backlog.* 202
Historia de usuario. 202
Check list calidad. 204
Matriz de dependencias. 204
Sprint Review. 205
Sprint Retrospective. 205

Sprint Dashboard. 205

Plan de *release.* 205

Solicitud de *release management.* 205

Solicitud de versionamiento. 206

Release notes. 206

Plan de calidad. 206

Propuesta de herramientas colaborativas (Tool Chain) 207

1. Ecosistema IBM (puro) 207

2. Ecosistema Open (puro) 208

3. Ecosistema Mixto 210

Métricas de los equipos Scrumban 214

Definición de objetivos 214

Definición de indicadores 214

1. Continuidad en el flujo de necesidades. 215

2. Tiempo de ciclo. 217

3. Efectividad de la mejora continua. 218

Buenas prácticas ágiles por implementar 220

Técnicas de estimación 220

Pivote. 220

Planning poker. 221

Estimación por camisetas. 222

Peer review o revisión de pares 223

Implementación de la propuesta 225

Plan de implementación 225

Beneficios esperados 225
Análisis Costo -Beneficio 231
Costos de la propuesta 233
Costo de mano de obra. 233
Costo de materiales de capacitación. 235
Costo de herramientas colaborativas. 236
Costo Ecosistema IBM 236
Costo Ecosistema Open Puro. 237
Costo Ecosistema Mixto Opción 1 y 2. 238
Costos totales plan piloto fábrica de Integration 239
Posibles limitaciones 242
Conclusiones y recomendaciones finales 243
Glosario 244
Referencias 248
Apéndices 251
Apéndice No 1. Guía para conversatorio *personal Integration* 251
Apéndice No 2. Instrumentos para evaluar el nivel de madurez de las prácticas de desarrollo de *software* 253
Apéndice No 3. Resultados de la evaluación de madurez en prácticas de desarrollo por equipo y práctica 256
Apéndice No 4. Historia de usuario 263
Apéndice No 5. *Check list* de Calidad 264
Apéndice No 6. Matriz de dependencias 265
Apéndice No 7. *Sprint review* 266
Apéndice No 8. *Sprint retrospective* 267

Apéndice No 9. *Sprint dashboard* 268

Apéndice No 10. Plan de *release* 269

Apéndice No 11. Solicitud de *release* 270

Apéndice No 12. Solicitud de versionamiento 271

Apéndice No 13. *Release notes* 272

Apéndice No 14. Plan de pruebas 273

Apéndice No 15. Análisis comparativo de herramientas colaborativas 274

Apéndice No 16. *Peer review* 279

Apéndice No 17. Plan de implementación 292

Apéndice No 18. Costo de mano de obra 294

Apéndice No 19. Costo de materiales de capacitación 295

Índice de figuras

Figura N.º 1 Diagrama Ishikawa 39

Figura N.º 2 Diagrama de Gantt 40

Figura N.º 3 Modelo de la Cascada 45

Figura N.º 4 Metodología tradicional versus ágil 47

Figura N.º 5 Ciclo Scrum 50

Figura N.º 6 Burndown Chart 58

Figura N.º 7 Pizarra Kanban 60

Figura N.º 8 Tarjetas Kanban 62

Figura N.º 9 Fases enfoque mixto (cualitativo integral) 73

Figura N.º 10 Procedimiento general del análisis para el diseño mixto 81

Figura N.º 11 WBS (*Work Breakdown Structure*) 82

Figura N.º 12 Diagrama de Gantt 83
Figura N.º 13 Diagrama de proceso general área de Integration 86
Figura N.º 14 Diagrama de flujo Equipo Integration Core 90
Figura N.º 15 Diagrama de flujo Equipo Business Automation 95
Figura N.º 16 Diagrama de flujo Equipo SO 99
Figura N.º 17 Cálculo de muestras para poblaciones finitas 101
Figura N.º 18 Madurez actual práctica de levantamiento de requerimientos Equipo IC 105
Figura N.º 19 Madurez actual práctica de *Release management* Equipo IC 107
Figura N.º 20 Madurez actual práctica de Aseguramiento y Control de calidad 109
Figura N.º 21 Madurez actual práctica de levantamiento de requerimientos Equipo BA 112
Figura N.º 22 Madurez actual práctica de *Release management* Equipo BA 114
Figura N.º 23 Madurez actual práctica de Aseguramiento y Control de Calidad Equipo BA 115
Figura N.º 24 Madurez actual práctica Levantamiento de requerimientos Equipo SO 119
Figura N.º 25 Madurez actual práctica *Release management* Equipo SO 121
Figura N.º 26 Madurez actual práctica de Aseguramiento y Control de Calidad Equipo BA 123
Figura N.º 27 Diagrama de Ishikawa 126
Figura N.º 28 Apartados de la propuesta 142
Figura N.º 29 Diagrama de Proceso Scrumban 144
Figura N.º 30 Estructura de la fábrica 146
Figura N.º 31 Diagrama de flujo solicitudes nuevas 151
Figura N.º 32 Diagrama de flujo levantamiento de la necesidad 152
Figura N.º 33 Diagrama de flujo Planning 153
Figura N.º 34 Tablero Kanban – Carril Solicitudes nuevas - Carril *fast line* 154
Figura N.º 35 Diagrama de flujo carril *fast line* 155

Figura N.º 36 Diagrama de flujo Gestión de requerimientos 161
Figura N.º 37 Eventos o ceremonias para el seguimiento 161
Figura N.º 38 Diagrama Daily Scrum Meeting 163
Figura N.º 39 Diagrama de flujo *Sprint review* 164
Figura N.º 40 Diagrama de flujo Retrospectiva 166
Figura N.º 41 Diagrama de gestión de la calidad 170
Figura N.º 42 Diagrama de flujo *Release management* 175
Figura N.º 43 Estructura de versionamiento Integration 176
Figura N.º 44 Ejemplo *branch* de desarrollo 177
Figura N.º 45 Ejemplo uso de nomenclatura en los *branch* 178
Figura N.º 46 Subproceso solicitud de *Release management* 179
Figura N.º 47 Diagrama de flujo Gestión de Carpetas para Nuevos Desarrollos 180
Figura N.º 48 Diagrama de flujo manejo de *branch* 181
Figura N.º 49 Diagrama de flujo subproceso gestión de *release* QA 182
Figura N.º 50 Diagrama de flujo subproceso gestión de *release* 183
Figura N.º 51 Diagrama de flujo subproceso gestión de garantía 184
Figura N.º 52 Ciclo de desarrollo de la fábrica 207
Figura N.º 53 Tool chain Ecosistema IBM 208
Figura N.º 54 Tool Chain Ecosistema Open 210
Figura N.º 55 Tool Chain Mixto 1 212
Figura N.º 56 Tool Chain Ecosistema Mixto Opción 2 213

Índice de tablas

Tabla N.º 1 Descripción de variables: Conceptual, operativo e instrumental 76

Tabla N.º 2 Descripción de instrumentos 79

Tabla N.º 3 Hallazgos Equipo IC 87

Tabla N.º 4 Hallazgos Equipo BA 92

Tabla N.º 5 Hallazgos Equipo SO 97

Tabla N.º 6 Madurez actual práctica de levantamiento de requerimientos Equipo IC 104

Tabla N.º 7 Madurez actual práctica de *Release management* Equipo IC 106

Tabla N.º 8 Madurez actual práctica de aseguramiento y control de calidad Equipo IC 108

Tabla N.º 9 Madurez actual práctica de levantamiento de requerimientos Equipo BA 111

Tabla N.º 10 Madurez actual práctica de *Release management* equipo BA 113

Tabla N.º 11 Madurez actual práctica de aseguramiento y control de calidad Equipo BA 115

Tabla N.º 12 Madurez actual práctica Levantamiento de Requerimientos Equipo SO 117

Tabla N.º 13 Madurez actual práctica de *Release management* Equipo SO 120

Tabla N.º 14 Madurez actual práctica de aseguramiento y control de calidad Equipo SO 122

Tabla N.º 15 Tabla Multivoto 131

Tabla N.º 16 Resumen de votación de todos los miembros 132

Tabla N.º 17 Votos totalizados 133

Tabla N.º 18 Ponderaciones por Factor 135

Tabla N.º 19 Matriz EFI 136

Tabla N.º 20 Características de pruebas 168

Tabla N.º 21 Ecosistema Mixto opción 1 211

Tabla N.º 22 Opción 2 Ecosistema Mixto 212

Tabla N.º 23 Propuesta de objetivos para las células en Scrumban 215

Tabla N.º 24 Continuidad en el flujo de necesidades 216

Tabla N.º 25 Indicador tiempo de ciclo 217

Tabla N.º 26 Indicador efectividad de la mejora continua 218

Tabla N.º 27 Métricas prácticas ágiles 219

Tabla N.º 28 Plan de ejecución piloto 225

Tabla N.º 29 Beneficios esperados 226

Tabla N.º 30 Beneficios esperados 233

Tabla N.º 31 Mano de obra plan piloto 235

Tabla N.º 32 Lista de materiales necesarios capacitación plan piloto 235

Tabla N.º 33 Costos escenario Ecosistema IBM 237

Tabla N.º 34 Costo total Ecosistema IBM 237

Tabla N.º 35 Costos escenario Ecosistema Open 238

Tabla N.º 36 Costo escenario Ecosistema Mixto Opción 1 239

Tabla N.º 37 Costo escenario Ecosistema Mixto Opción 2 239

Tabla N.º 38 Costo total piloto fábrica de Integration Ecosistema IBM 240

Tabla N.º 39 Costo total piloto fábrica de Integration Ecosistema Open 240

Tabla N.º 40 Costo total piloto fábrica de Integration Ecosistema Mixto Opción 1 240

Tabla N.º 41 Costo total piloto fábrica de Integration Ecosistema Mixto Opción 2 241

Tabla N.º 42 Relación Costo – Beneficio Escenarios propuestos 241

Dedicatoria

A mi **hija**, mi fuente de
inspiración y motivación.

A **Max, Dany y Raquel,** por su amor y ternura para con nosotras.

A mis **padres**, por todo su apoyo para lograr mis metas,
por darme motivación y sus manos cuando más lo necesité.

Agradecimientos

Como cierre de mis estudios universitarios para llegar a una de las metas que me he propuesto: la Licenciatura en Ingeniería Industrial, deseo agradecerle ante todo a mi familia, por su apoyo incondicional en este proceso.

Doy mi agradecimiento a los profesores de la carrera de Ingeniería Industrial por su labor y por dedicar su tiempo hacia los estudiantes con paciencia y sabiduría, en especial a los profesores Alejandro Leiva y Freddy Hernández, quienes me han guiado en este proyecto de investigación.

A la empresa GBM y el área de Integration, por el apoyo y la confianza para realizar este proyecto y a todas las personas que han contribuido al logro de esta meta.

RESUMEN EJECUTIVO

Este proyecto fue desarrollado en la empresa GBM de Costa Rica S.A., localizada en el Parque Comercial Lindora. El objetivo del estudio es definir una propuesta metodológica que le permita al área de Integration y a GBM innovar la forma de hacer sus procesos, determinando los insumos necesarios para el desarrollo de una metodología de trabajo que involucre nuevos procesos, herramientas, roles y funciones, permitiendo poseer mayor visibilidad, control y trazabilidad de todas las actividades que giran alrededor del ciclo de desarrollo de requerimientos.

Mediante un análisis de la empresa y del departamento como tal, se determinó como principal necesidad de cambio que no hay procesos estandarizados ni metodología de trabajo clara, donde los líderes de los equipos realizan su labor de forma informal, generando falta de control de los desarrollos. Se analizó el área en estudio en su interior, mediante la matriz EFI, de la cual se obtuvo como resultado un coeficiente de 1, lo que indica que el área está muy por debajo de la media (2,5), ubicándola como un área que presenta grandes debilidades en comparación con sus fortalezas.

La propuesta de solución consistió en el descubrimiento de los procesos y metodología que mejor se adapten a la naturaleza de los trabajos del área, definición de una nueva estructura, el diseño de plantillas y herramientas colaborativas que soporten los procesos propuestos y el descubrimiento de métricas para una metodología Scrumban. Se recomienda la implementación del escenario 1 Ecosistema IBM, o bien, la opción 1 del escenario 3 Ecosistema Mixto, como definición de las herramientas colaborativas necesarias para lograr con éxito la implementación de la metodología.

Se propone un plan piloto con una duración estimada de 89,5 días, el mismo requiere de un monto de inversión por parte de GBM, el cual incluye la inversión en horas para la preparación y ejecución de las capacitaciones, los materiales requeridos para las mismas, así como el tiempo que invertirán los diferentes *coach* y el equipo como tal durante el periodo de ejecución del piloto.

CAPÍTULO I. INTRODUCCIÓN

En el campo de la ingeniería, cuando se desarrolla *software* es de gran importancia aplicar metodologías de trabajo que permitan medir el avance, monitoreando el tiempo, costo, alcance y calidad de su realización. A lo largo del tiempo, se han utilizado formas de trabajo tradicional de tipo cascada o *waterfall*, las cuales plantean la realización del proyecto contemplando las siguientes fases: identificación de la necesidad, análisis, diseño, desarrollo y pruebas, hasta la entrega del producto final.

Esta forma de trabajar no siempre aseguró el éxito de los proyectos y fue desde 1986, cuando se da como metodología alternativa a la tradicional el primer registro de prácticas ágiles bajo el término Scrum, como analogía a las estrategias del rugby como deporte para el desarrollo de producto en el artículo *The New Product Development Game* publicado en el Harvard Business Review. Desde ese momento, ya se visualizaban resultados positivos al tener equipos auto-organizados, manejar la superposición de fases, ejercer un control más sutil del proyecto y mantener un aprendizaje continuo en empresas como Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M, Xerox y Hewlett-Packard.

Sin embargo, no fue sino hasta el año 2001 que 17 agilistas se reunieron a definir y firmar el Manifiesto Ágil con los cuatro valores y 12 principios, los cuales marcan la diferencia entre las metodologías tradicionales y las ágiles, que supone deben regir en todo momento en el desarrollo de los proyectos.

Estudios recientes realizados por Standish Group revelan que, en general, aquellos proyectos que se trabajaron bajo metodologías ágiles lograron un 39 % de éxito frente a solo un 9 % de los que fueron desarrollados bajo metodologías tradicionales. Este comportamiento se mantiene al analizar los proyectos por tamaño ya sean grandes, medianos o pequeños. Así mismo, reitera que el involucramiento de los interesados y usuarios en todo el proceso, el trabajo en equipo, la optimización y mejora continua, el personal capacitado, la definición de prácticas ágiles a seguir y el dominio de estas siguen siendo factores de éxito.

Por esta razón, como respuesta a las necesidades de innovación, el actual proyecto propone, mediante una revisión de la planeación, control y ejecución de proyectos, el diseño de una metodología que contemple los principios ágiles en la gestión de proyectos de desarrollo de *software*, la cual permita entregas iterativas e incrementales (mini-proyectos conformados por las

fases del desarrollo: requerimientos, diseño, desarrollo, *testing*) bajo medios de comunicación permanentes entre desarrolladores e interesados, apoyándose en herramientas de desarrollo colaborativo que permitan automatizar procesos y un eficiente seguimiento del proyecto, logrando la retroalimentación necesaria de todos los involucrados. Esta última limitada en los modelos tradicionales después de finalizar la fase de requerimientos.

Además, se definen como objetivos específicos de este trabajo la investigación de prácticas ágiles y herramientas colaborativas, análisis del estado de la madurez en prácticas de desarrollo de *software*, así como la definición metodológica que incluirá herramientas, plantillas, procesos, procedimientos y plan de trabajo, los cuales podrán ser utilizados por la organización en su etapa de implementación.

Se escoge General Business Machine (GBM) para hacer la investigación, ya que es una organización que cuenta con la apertura y motivación necesaria para innovar en la forma en que hoy en día se gestionan y ejecutan los proyectos, además, se evidencia más allá del interés por innovar, la necesidad de un cambio para seguir estando a la vanguardia de las exigencias del mercado en el que compite.

Generalidades de la empresa

A principios de los 90, el mercado de tecnología de información sufrió drásticos cambios a nivel mundial, de los cuales IBM no estuvo ajeno. En esta década, las empresas sustituyeron sus plataformas centralizadas de *mainframes*, por las de servidores de mediano rango enlazados entre sí, para el manejo de los diferentes procesos. En esta década, la reingeniería de procesos tenía un papel predominante dentro del mundo de los negocios.

Al igual que las demás corporaciones multinacionales, IBM, durante los tres primeros años de la década de los 90, tuvo que revisar sus esquemas de negocios y rediseñar sus procesos, líneas de producto y esquemas de distribución. Esto trajo como consecuencia la necesidad de buscar socios o aliados de negocio en algunos mercados, donde por razones económicas, operativas y geográficas, pudieran ser manejados de una manera más eficiente como lo requerían los nuevos tiempos y modelos de distribución de la industria.

Es de esta manera como surge, a fines de 1991, una alianza con un grupo de empresarios centroamericanos para el manejo de la marca IBM en la región Centroamericana y el Caribe, y se formó así la corporación GBM (General Business Machines), donde IBM mantiene una participación accionaria. La nueva alianza cuenta con el acceso a sus vastos recursos técnicos y de soporte, para garantizar de esta manera una relación transparente a los clientes de IBM en la región.

De igual forma, el área de Centroamérica y el Caribe estaba experimentando un profundo proceso de estabilización política y económica, que se reflejaría en un urgente esfuerzo de modernización técnico-administrativa en todas las áreas. Dentro de este proceso, dos factores cumplirían un papel determinante: la cercanía geográfica y los estrechos vínculos comerciales del territorio con el mercado de los Estados Unidos. La conexión con Norteamérica hace que el mercado centroamericano se encuentre hoy entre los más actualizados y modernos de toda Latinoamérica.

GBM ha asimilado lo mejor de la cultura organizacional de su socio IBM, lo cual le ha permitido contar con las últimas plataformas tecnológicas disponibles para el mercado, a través de la introducción de productos y servicios orientados a satisfacer las cambiantes necesidades, apoyados por el entrenamiento continuo del personal técnico y de apoyo, a fin de brindar un servicio de calidad mundial a todos los clientes de este mercado.

La alianza estratégica con IBM representa una garantía de acceso sin restricciones a la más completa línea de productos en el mundo, así como a sus vastos recursos de soporte. Esta estrecha asociación permite a GBM proyectar el futuro con la base del conocimiento de las tendencias más estratégicas de la industria.

El dinámico proceso de cambio que experimenta la industria informática, en el que dramáticos avances técnicos se producen cada día, exige que las empresas proveedoras desarrollen un catálogo extenso de productos, con plataformas alternativas que incluyan una variedad de tecnologías complementarias y los servicios de implantación y soporte correspondientes. Solo así se estará en condiciones de responder al conjunto de necesidades del cliente, de configurar e integrar todos los sistemas que en definitiva representarán una verdadera solución para el usuario.

GBM está basada en la asociación de un número de líderes de negocios con un equipo gerencial de vasta experiencia en la industria y cuenta con un cuerpo de profesionales que acumula una capacidad técnica sin paralelo en el territorio. Esta combinación le permite operar como una empresa auténticamente local, con comprensión de las circunstancias internas de cada país, a la vez que capitaliza la sinergia resultante de su cubrimiento regional (GBM es el único proveedor de tecnología con presencia regional) y la masa crítica combinada del conjunto de los mercados.

GBM es distribuidor exclusivo de IBM en Centroamérica, Panamá y República Dominicana y según acuerdo entre las partes, su relación con IBM no es restrictiva. Si bien basa sus soluciones en la plataforma técnica IBM y mantiene una coherencia estratégica, GBM representa, además, de forma oficial y con pleno apoyo, marcas líderes de la industria y complementarias de la línea IBM, como CISCO y DIEBOLD, con el fin de ofrecer a los clientes una solución más completa. GBM tiene como principio mantener el nivel de excelencia en el soporte que brinda a cada marca que representa. En consecuencia, la capacitación de los técnicos en cada una de estas líneas es un prerrequisito ineludible para cualquier producto que GBM decida comercializar en la región.

GBM es líder en Centroamérica y el Caribe en integración de soluciones de tecnología de información para sus clientes, con el fin de agregar valor a sus negocios, derivando de ahí los resultados financieros esperados. Las principales líneas de producto que se manejan en GBM son:

Servicios: servicio técnico y mantenimiento, datacenter, educación, servicios gestionados, *software* services.

Hardware: servidores, computadores personales, productos de redes, puntos de venta, entre otros.

Software: *middleware*, aplicaciones, Business Inteligence, Core banking, SAP.

Consultoría: en las líneas de Change Management, BPO, entre otras.

Visión

"Ser los mejores proveedores de soluciones de TI de nuestros clientes para mejorar su competitividad, con el propósito de duplicar el negocio y la rentabilidad en 5 años, en un clima organizacional óptimo, innovador y colaborativo con las comunidades donde operamos" (GBM, 2018,Visión).

Misión

"Integrar la tecnología en soluciones de valor agregado que satisfagan las expectativas de nuestros clientes, a través de profesionales calificados y comprometidos, con metodologías, productos y servicios de clase mundial" (GBM,2018, Misión).

Valores

Confiabilidad: Ser honestos, íntegros y leales, ejecutando nuestros compromisos con alta calidad, precisión y puntualidad.

Coraje: Sinónimo de atrevimiento. Ser los más genuinos, persistentes y productivos.

Disciplina: Observancia y cumplimiento de las reglas y compromisos.

Transparencia: Ser claro, evidente, sin duda ni ambigüedad. (GBM, 2018,Valores).

Planteamiento del problema

Ante la amenaza de perder competitividad y la exigencia del mercado en cuanto a últimas tendencias en la gestión y ejecución de proyectos, GBM se ve en la necesidad de innovar en la forma de brindar soluciones, lo que da pie a la realización del presente proyecto sobre diferentes maneras de hacer los procesos para lograr mejores resultados que beneficien tanto a la empresa como a los clientes.

Hasta el día de hoy, GBM ha desarrollado *software* guiado bajo las metodologías tradicionales de gestión y desarrollo de proyectos, donde los procesos y herramientas, documentación extensiva, negociación contractual y seguir un plan de forma rígida están limitando de una u otra forma la mejora continua, la satisfacción de los clientes actuales y la atracción de nuevos clientes a la cartera. También se ha dificultado la estandarización de la misma en las diferentes áreas de trabajo, lo que está evidenciando problemas de rendimiento y satisfacción de los clientes, así como desperdicios innecesarios.

Ante esto, se presenta la siguiente interrogante: ¿Cuáles metodologías y cuáles herramientas colaborativas se deben implementar para lograr una gestión donde las estimaciones de los proyectos y la toma de decisiones sucedan bajo menores grados de incertidumbre, con un mayor nivel de calidad y un desgaste menor de los equipos de trabajo, considerando la participación continua de los clientes para una mayor satisfacción de sus expectativas?

Objetivo general

Diseñar una guía metodológica basada en prácticas ágiles adaptables a herramientas colaborativas, para la gestión y ejecución de proyectos de *software* que permitan a la compañía la innovación y consolidación como líderes en el mercado de desarrollo de *software*.

Objetivos específicos

- Documentar la situación actual tanto a nivel metodológico como de procesos del área en estudio.
- Determinar la propuesta metodológica idónea y las herramientas colaborativas necesarias para implementarla.

- Diseñar los procesos principales, métricas y plantillas de trabajo mínimas necesarias para poner en marcha la metodología propuesta.

Justificación

Estudios sobre las mejores prácticas en desarrollo de proyectos de *software* indican que los mercados actuales se están caracterizando por la implementación de cambios en la forma de hacer sus procesos, realizando desarrollos de *software* bajo metodologías ágiles tanto a nivel de gestión como de prácticas ágiles para las pruebas de calidad, *release management* (versionamiento) y desarrollo de las aplicaciones.

Las metodologías ágiles son marcos de trabajo que definen lineamientos para ejecutar proyectos de diferentes tipos. De tal forma que la entrega del producto sea incremental, garantizando la calidad, en tiempos de entrega relativamente cortos, logrando transparencia, colaboración en equipo y puesta en práctica de métodos que ayudan a los desarrolladores a adaptarse con rapidez a cambios y nuevos imprevistos.

Ante esto, el presente estudio busca definir una metodología que logre orientar los desarrollos hacia un enfoque ágil dirigiendo todos los esfuerzos hacia la innovación, con el fin de que la organización pueda entrar a competir en un agresivo mercado de desarrollo de *software*. Además, como beneficios colaterales, se pretende tratar de solventar oportunidades de mejora que los métodos tradicionales han dejado de lado a lo largo del tiempo, entre los que están:

- Equipos de trabajo motivados, auto-organizados y proactivos.

- Satisfacción de las expectativas del cliente desde etapas tempranas de los proyectos.

- Identificación de riesgos como una responsabilidad de todos en el equipo.

- Flexibilidad para la gestión de control de cambios.

-Inclusión del control de calidad desde etapas tempranas del desarrollo.

Antecedentes

A la fecha, la organización se encuentra desarrollando sus primeras iniciativas en el diseño de metodologías ágiles para la gestión de proyectos de desarrollo de *software*.

Proyecciones

1- Marco teórico de las prácticas ágiles y herramientas colaborativas más utilizadas en compañías de desarrollo de *software*, identificando aquellas que más se adaptan a la necesidad planteada.

2- Resultados del análisis de madurez metodológico como guía para la definición de la propuesta.

3- Metodología de trabajo ágil, de forma tal que se definan las herramientas, los procesos, roles y plantillas necesarias para su implementación.

4- Plan de implementación de la metodología y prácticas ágiles de desarrollo.

CAPÍTULO II. MARCO TEÓRICO

El marco teórico que se desarrolla a continuación permite conocer los conceptos básicos por utilizar en el presente proyecto, los cuales son necesarios para su entendimiento y comprensión. Primero, se explican las herramientas que serán utilizadas para realizar el análisis de la situación actual, luego se explicarán términos básicos de estadística, seguidos por una definición de metodologías de trabajo tradicionales como ágiles. Una vez expuestos estos temas, se revisan las definiciones a alto nivel de las herramientas colaborativas que se están planteando para el diseño de la metodología ágil.

A la vez, se documenta una breve explicación sobre el diagrama de Gantt, herramienta que apoyará el proceso de planeación. Una planeación correcta y ordenada permitirá llevar un proceso adecuado de investigación y desarrollo.

Herramientas de análisis situación actual

Ingeniería de métodos

De acuerdo con lo que expone Palacios (2009), la ingeniería de métodos consiste en decidir dónde agrega valor el ser humano en el proceso de convertir materias primas en productos terminados y en decidir cómo puede una persona desempeñarse de forma efectiva en las tareas que se le asignen, para lograr de esta forma el desempeño efectivo, tomando en cuenta que los costos de contratar, capacitar y entrenar a una persona, cada vez son más altos.

Además, se manifiesta lo siguiente: “Es evidente que el ser humano es y será por mucho tiempo, una parte importantísima del proceso de producción en cualquier tipo de planta. Pero también es cierto, que su óptimo aprovechamiento dependerá del grado de utilización de su inteligencia, de su potencial de ingenio y creatividad” (Palacios, 2009, p. 30).

Considerando y tomando en cuenta la recomendación de Palacios (2009), se utilizará en la presente investigación el estudio de los métodos para prever la metodología que mejor se adapte según los roles, herramientas y procedimientos requeridos y potencialmente posibles de alcanzar dentro de la organización en estudio. A continuación, se detallan las herramientas de análisis que serán utilizadas en el presente estudio.

Flujograma.

Según explica Palacios (2009), un flujograma sirve para representar procesos que exigen decisiones describiendo de forma individual el procedimiento para cierta porción del sistema. Los flujogramas sirven como guía en la ejecución de los procesos, mostrando la secuencia lógica y dinámica del trabajo; permitiendo el entendimiento de las unidades que intervienen en ella y el proceso que se describe a través del procedimientos. De acuerdo con lo expuesto, se recomienda organizar las diferentes fases del proceso en un diagrama de flujo, antes de iniciar con la elaboración de los procedimientos (Palacios, 2009).

Para mapear procesos existen diversas herramientas que ayudan a capturar y automatizar los procedimientos de negocios y que ayudan en la tarea de colaborar con la toma de decisiones, como, por ejemplo, IBM BlueWorksLive, que pertenece al marco de aplicaciones de IBM SmartCloud y Visio, el cual forma parte del marco de aplicaciones Microsoft.

IBM BlueWorksLive.

BlueworksLive es una herramienta de gestión de procesos empresariales, lanzada al mercado por IBM en 2010, basada en *cloud computing*, que permite descubrir, diseñar, automatizar y gestionar procesos empresariales. De acuerdo con IBM (2017), es fácil de usar y accesible en cualquier lugar con un navegador, intenta integrar los principios de redes sociales con el modelado de procesos de negocio y cubre también dos áreas de trabajo simples:

- Área de Comunidad, en la que se identifica el proceso.
- Área de Trabajo, donde los usuarios pueden desarrollar aplicaciones de procesos basados en Java.

Es una herramienta centralizada y colaborativa que mantiene todos los activos relacionados con procesos en un mismo lugar, así todas las personas involucradas en el proceso están en la misma página (IBM, 2017).

Existen dos niveles de detalle, como menciona IBM (2017), uno es un mapa de descubrimiento que contiene solo los hitos y las actividades de un proceso, el otro es un diagrama de proceso completo donde se pueden representar las funciones de las personas o departamentos que participan en los procesos. La aplicación también permite a los usuarios tener en cuenta

varios problemas relacionados con los procesos ya capturados, la captura de los propios problemas, así como su descripción, gravedad y la frecuencia de ocurrencia.

Diagrama Ishikawa.

Según Acuña (2012), el diagrama de Ishikawa, también conocido como diagrama de espina de pescado, sirve para obtener información sobre las características de calidad generadas en la fabricación del producto asociadas a un proceso o a un producto y ordenarlas en categorías. Agrega que existen cuatro tipos de diagramas de Ishikawa: el diagrama de procesos, el diagrama de producto, el diagrama de características y el diagrama de factores. Para efectos de esta investigación, se utilizará el diagrama de factores, en el que se colocan todas las características asociadas a los factores de la calidad, tales como capital, tecnología, material, recurso humano, mercado, administración y métodos.

Acuña (2012) especifica el procedimiento para construir cualquiera de los diagramas de Ishikawa de la siguiente manera:

1. Elegir el producto que será objeto de estudio. Esto se debe hacer sobre la base de las quejas recibidas de los clientes y los informes de producción que reflejen condiciones desfavorables en el comportamiento de la calidad del producto.

2. Colocar la palabra producto en el extremo derecho de una flecha horizontal

3. Hacer una lista de todas las características de calidad que se generan. Esta lista se efectúa para cada parte del producto, cada etapa del proceso, para cada factor o en forma general. En esto, se debe tomar en cuenta al operario y a los inspectores experimentados, con el fin de no dejar por fuera aquellas características que tengan una periodicidad muy irregular o muy espaciada y que, por lo tanto, puede que no se detecten en el periodo de observación. De ninguna manera esta lista debe ser el producto de un análisis individual, sino de trabajo en equipo.

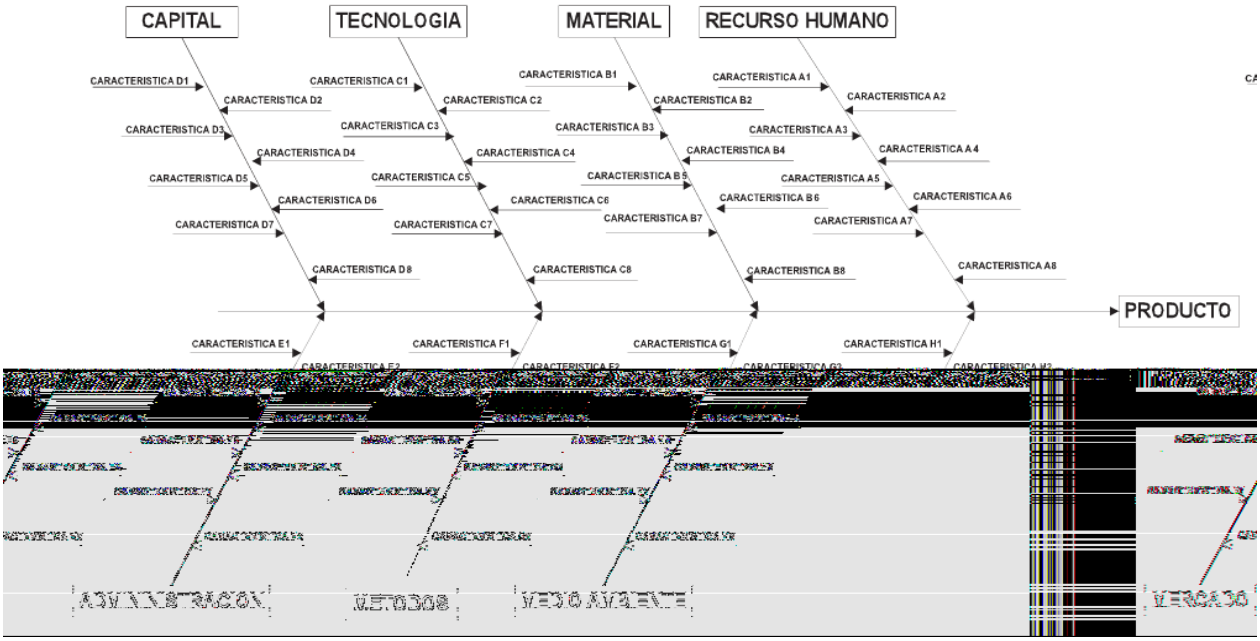
4. Ordenar la información en forma secuencial, de acuerdo con las partes que componen al producto, las etapas que conforman el proceso, el orden de los factores de calidad o el orden de las características principales.

5. Dibujar las flechas diagonales (ramas principales) sobre las que se representarán las partes del proceso, las partes del producto, los factores o las características de calidad.

6. Dibujar subramas y anotar en ellas las características de calidad. Para el diagrama de factores, se anotan las características de calidad asociadas a cada factor. Se pueden anotar las causas sobre pequeñas ramas de cada subrama para todos los diagramas. (Acuña, 2012, p. 207)

Figura N.º 1

Diagrama Ishikawa



Nota: Acuña (2012).

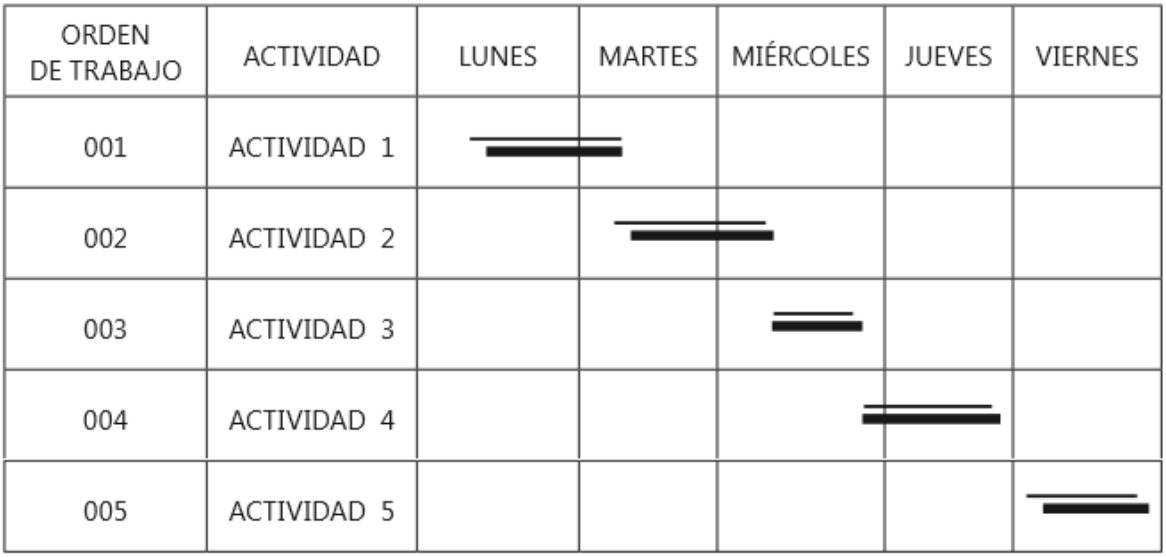
Diagrama de Gantt.

Según Heizer & Render (2009), los diagramas de Gantt se deben a Henry Gantt, quien desarrolló la herramienta a finales del siglo XIX. El diagrama de Gantt es una ayuda visual para determinar las cargas de trabajo y la programación de una producción, así como los tiempos ociosos de máquinas y departamentos, las gráficas muestran el uso de los recursos.

Una gráfica de Gantt de programación se usa para vigilar el avance de los trabajos indicando cuáles tareas están a tiempo y cuáles están adelantadas o atrasadas (Heizer & Render, 2009, p. 608). Según la explicación dada por Palacios (2009), este diagrama consta de una gráfica de doble entrada donde las filas representan máquinas, personas, departamentos o recursos que sean necesarios para cumplir una tarea. Las columnas definen los períodos en horas, días, semanas o meses de la siguiente manera:

Figura N.º 2

Diagrama de Gantt



Nota: Palacios (2009).

Tabla multivoto

La tabla multivoto es definida como un método para calificar problemas, causas de problemas o limitaciones de un proceso de mejoramiento continuo (Acuña, 2012). Una clasificación inicial se somete a votación de un grupo de no menos de 10 personas, a fin de que cada una de ellas exprese su opinión con respecto a la criticidad del problema, causa o limitación, votando por cada una de ellas a través de la asignación de un número que va de 1 a 5, siendo 1 de mayor importancia y 5 de menor importancia. El 3 representa una opinión intermedia (Acuña, 2012, p. 79).

El procedimiento que estipula Acuña (2012) es el siguiente:

1. Construir una tabla con un listado de todas las causas, problemas o limitaciones de la organización, numeradas en forma consecutiva y que pueden ser el producto de un diagrama de Ishikawa. Este listado está constituido por los reglones de la tabla. Luego, insertar cinco columnas numeradas del 1 al 5.

2. Repartir la tabla a cada uno de los miembros del grupo, los cuales son idóneos para ejecutar la asignación de voto. Cada miembro coloca una “x” en la columna que corresponde a su opinión de la causa, problema o limitación en análisis.

3. Generar la tabla de frecuencias en donde se anota el número de veces que se votó por cada columna.

4. Obtener el voto ponderado multiplicando el valor obtenido en 4 por el valor de la columna (1, 2, 3, 4, 5).

5. Sumar los puntos para cada idea y seleccionar los de más bajo puntaje como los factores de análisis prioritario. (p.80)

Matriz de Evaluación del Factor Interno (EFI).

La Matriz de Evaluación del Factor Interno (EFI) es una herramienta para la formulación de una estrategia, resumiendo y evaluando las fortalezas y debilidades en las áreas funcionales de la empresa. Además, proporciona una base para identificar y evaluar las relaciones entre estas áreas (David, 2008). Según David (2008), una matriz EFI se elabora en cinco pasos:

1. Elaborar primero una lista de las fortalezas y después de las debilidades. Se debe ser lo más específico posible, usando porcentajes, índices y cifras comparativas.

2. Asignar un valor que vaya de 0.0 (sin importancia) a 1.0 (muy importante), a cada factor. El valor asignado a determinado factor indica la importancia relativa del factor para que sea exitoso en la industria de la empresa. Sin importar si un factor clave es una fortaleza o una debilidad interna, los factores considerados como aquellos que producen los mayores efectos en el rendimiento de la empresa deben recibir los valores más altos. La sumatoria de todos los valores debe ser igual a 1.0.

3. Asignar una clasificación de uno a cuatro a cada factor, para indicar si dicho factor representa una debilidad mayor (clasificación de uno), una debilidad menor (clasificación de dos), una fortaleza menor (clasificación de tres) o una fortaleza mayor (clasificación de 4). Se observa que las fortalezas deben recibir una clasificación de tres o cuatro. De este modo, las clasificaciones se basan en la empresa, mientras que los valores del paso se basan en la industria.

4. Multiplicar el valor de cada factor por su clasificación para determinar un valor ponderado para cada variable.

5. Sumar los valores ponderados de cada variable para determinar el valor ponderado total de la empresa.

Sin importar cuántos factores estén incluidos en una matriz EFI, el puntaje de valor total varía de 1.0 a 4.0, siendo el promedio de 2.5. Los puntajes de valor muy por debajo de 2.5 caracterizan a las empresas que son débiles internamente, mientras que los puntajes muy por arriba de 2.5 indican una posición interna sólida (David, 2008).

Estadística.

Variable.

Arias (2012) define variable como: “la característica o cualidad; magnitud o cantidad, que puede sufrir cambios, y que es objeto de análisis, medición, manipulación o control en una investigación” (p. 57).

Tipos de variables.

De acuerdo con Arias (2012), las variables se pueden clasificar según su naturaleza en cuantitativas, las cuales pueden ser continuas o discretas y las cualitativas, las cuales pueden ser dicotómicas y policotómicas. A su vez, Arias expone que, según el grado de complejidad, las variables pueden ser simples o complejas y según su función, se pueden clasificar en dependientes, independientes, intervinientes y extrañas. A continuación, se presentan las definiciones puntuales dadas por Arias (2012) según cada clasificación:

Según su naturaleza:

Cuantitativas.

Son aquellas que se expresan en valores o datos numéricos.

Discretas: son las que asumen valores o cifras enteras.

Continuas: son aquellas que adoptan números fraccionados o decimales.

Cualitativas.

También llamadas categóricas, son características o atributos que se expresan de forma verbal (no numérica), es decir, mediante palabras.

Dicotómicas: se presentan en solo dos clases o categorías.

Policotómicas: se manifiestan en más de dos categorías.

Según su complejidad:

Simples.

Las variables simples son las que se manifiestan directamente a través de un indicador o unidad de medida. No se descomponen en dimensiones.

Complejas.

Las variables complejas son aquellas que se pueden descomponer en dos dimensiones como mínimo. Luego se determinan los indicadores para cada dimensión.

Según su naturaleza:

Independientes.

Son las causas que generan y explican los cambios en la variable dependiente. En los diseños experimentales, la variable independiente es el tratamiento que se aplica y manipula en el grupo experimental.

Dependientes.

Son aquellas que se modifican por acción de la variable independiente. Constituyen los efectos o consecuencias que se miden y que dan origen a los resultados de la investigación.

Intervinientes.

Son las que se interponen entre la variable independiente y la dependiente, pudiendo influir en la modificación de esta última. En un diseño experimental puro, este tipo de variable debe ser controlada, con el fin de comprobar que el efecto es debido a la variable independiente y no a otros factores.

Extrañas.

También llamadas ajenas, son factores que escapan del control del investigador y que pueden ejercer alguna influencia en los resultados (Arias, 2012).

Técnica e instrumento.

Una técnica se refiere al procedimiento o forma particular de obtener datos o información. Para Arias (2012) existen dos tipos de técnicas: el diseño de investigación documental utilizando análisis documental o análisis de contenido y el diseño de investigación de campo, el cual puede utilizar la observación, encuestas, y entrevistas (Arias, 2012). Por otro lado, Arias (2012) define instrumento de recolección de datos como cualquier recurso, dispositivo o formato (en papel o digital), que se utiliza para obtener, registrar o almacenar información.

Para efectos de esta investigación y considerando lo expuesto por Arias (2012), se abarcan las dos técnicas: diseño de investigación documental y diseño de investigación de campos, utilizando como instrumentos la documentación almacenada y guía estructurada.

Por su parte, una encuesta es una técnica que pretende obtener información que suministra un grupo o muestra de sujetos acerca de sí mismos o en relación con un tema en particular. La entrevista, más que un simple interrogatorio, es una técnica basada en un diálogo o conversación "cara a cara", entre el entrevistador y el entrevistado acerca de un tema previamente determinado, de tal manera que el entrevistador pueda obtener la información requerida (Arias, 2012, p. 82).

Población.

Arias (2012) define la población objetivo como un conjunto finito o infinito de elementos con características comunes que serán estudiadas en la investigación.

Muestra.

La muestra es un subconjunto representativo y finito que se extrae de la población accesible. Una muestra representativa es aquella que, por su tamaño y características similares a las del conjunto, permite hacer inferencias o generalizar los resultados al resto de la población con un margen de error conocido (Arias, 2012, p. 84).

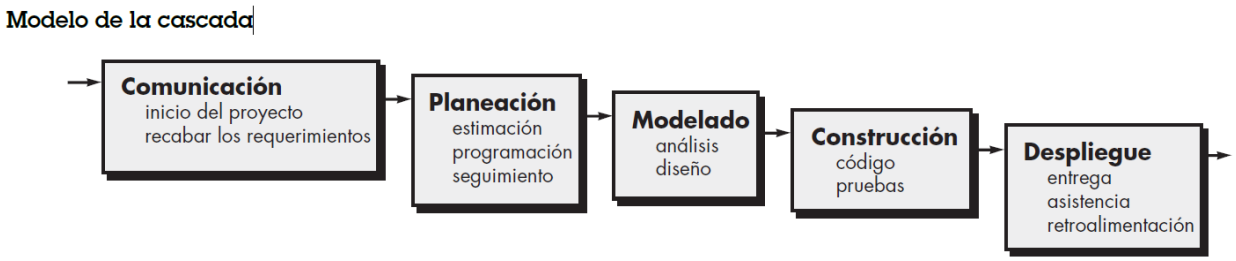
Para seleccionar la muestra. se conoce la técnica denominada muestreo, el cual puede ser probabilístico o aleatorio o no probabilístico.

Metodología tradicional: Método cascada o *waterfall*

Pressman (2010) define el modelo de la cascada como un enfoque sistemático y secuencial para el desarrollo del *software*, que da inicio con la identificación de los requerimientos por parte del cliente y avanza a través de planeación, diseño, construcción y despliegue, para concluir con la entrega y despliegue del *software* terminado (ver figura 3).

Figura N.º 3

Modelo de la Cascada



Nota: Pressman (2010).

Según Pressman (2010), el modelo de cascada suele ser efectivo cuando se tiene total claridad entendimiento y estabilidad de los requerimientos, cuando el trabajo desde la comunicación hasta el despliegue fluye en forma razonablemente lineal, agrega que estas características usualmente están presentes cuando deben hacerse adaptaciones o mejoras bien definidas a un sistema ya existente.

De acuerdo con Pressman (2010), el modelo de la cascada es el paradigma más antiguo de la ingeniería de *software*. Sin embargo, en los últimos años, se han evidenciado problemas o críticas sobre el uso de este modelo. Entre los problemas que en ocasiones surgen al aplicar el modelo de la cascada, Pressman (2010) menciona los siguientes:

1- Es raro que los proyectos reales sigan el flujo secuencial propuesto por el modelo. Aunque el modelo lineal acepta repeticiones, lo hace en forma indirecta. Como resultado, los cambios generan confusión conforme el equipo del proyecto avanza.

2. A menudo, es difícil para el cliente enunciar en forma explícita todos los requerimientos. El modelo de la cascada necesita que se haga y tiene dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos.

> 3. El cliente debe tener paciencia. No se dispondrá de una versión funcional del (de los) programa(s) hasta que el proyecto esté muy avanzado. Un error grande sería desastroso si se detectara hasta revisar el programa en funcionamiento. (Pressman, 2010)

Pressman (2010) asegura que, hoy en día, el desarrollo de *software* es acelerado y está sujeto a una corriente sin fin de cambios (en los requerimientos, funciones y contenido de información), por lo que considera que el modelo de cascada suele ser inapropiado para ese tipo de ambientes cambiantes. No obstante, sirve como un modelo de proceso útil en situaciones en las que los requerimientos son fijos y el trabajo avanza en forma lineal hacia el final.

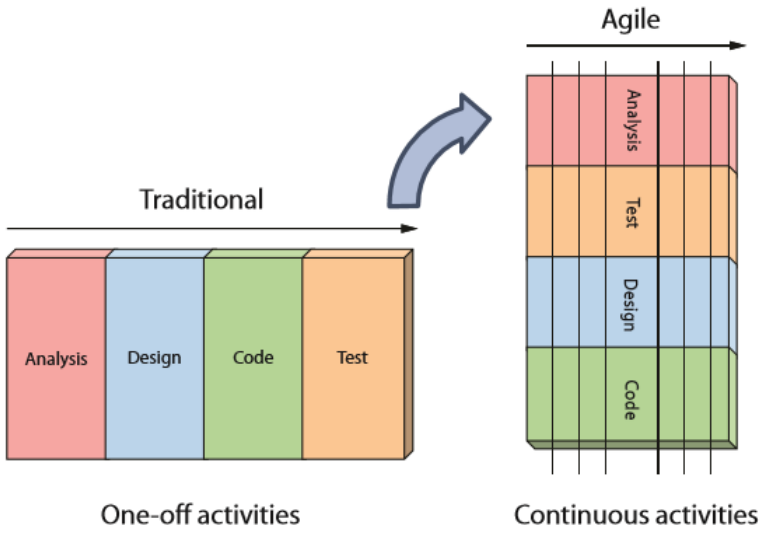
Metodologías ágiles.

Para Shore y Warden (2008), el desarrollo ágil es una filosofía. Es una forma de pensar sobre el desarrollo de *software,* en donde sus beneficio,s como la capacidad de lanzar *software* con mayor frecuencia, provienen de trabajar de manera diferente, no de trabajar más rápido.

Los métodos ágiles proponen la implementación de prácticas tales como el uso de control de versiones, el establecimiento de estándares de codificación y la entrega de demostraciones semanales a los interesados. La mayoría de estas prácticas han existido durante años, sin embargo, los métodos ágiles las combinan de maneras únicas, acentuando aquellas partes que respaldan la filosofía ágil, descartando el resto y mezclando nuevas ideas (Shore & Warden, 2008).

Figura N.º 4

Metodología tradicional versus ágil



Nota: Shore & Warden (2008).

Tal como muestra la figura, en metodologías tradicionales las fases o etapas tienen precedencia lineal, mientras que en el agilismo varias fases se pueden estar desarrollando de forma paralela.

Manifiesto ágil.

En marzo de 2001, un grupo de profesionales del *software*, expertos y críticos de los modelos de producción basados en procesos, fueron invitados por Kent Beck, para sostener un conversatorio sobre técnicas y procesos para desarrollar *software*. En esta sesión, se abordó la temática de *métodos ágiles* como alternativa a las metodologías formales tradicionales.

El grupo de expertos resumieron los principios sobre los que se basan los métodos alternativos en cuatro supuestos, conocidos como el manifiesto ágil:

> Estamos descubriendo formas mejores de desarrollar *software* tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:
>
> Los individuos e interacciones SOBRE los procesos y herramientas.
>
> El trabajo en *software* SOBRE la documentación completa.

La colaboración de los clientes SOBRE la negociación en los contratos.

La respuesta al cambio SOBRE el seguimiento de un plan. (Beck, 2001)

Principios del manifiesto ágil.

A continuación, se presentan los principios del manifiesto ágil, tomando literalmente las conclusiones expuestas en el documento del manifiesto (Beck, 2001):

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de *software* con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al Equipo de Desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Scrum.

De acuerdo con lo propuesto por Schwaber y Sutherland (2017) en su guía, se define Scrum como un marco de trabajo para el desarrollo y el mantenimiento de productos complejos. Mediante el cual las personas pueden afrontar problemas adaptativos, a la vez que entregan

productos con un máximo valor posible, de forma productiva y creativa. Entre sus características están:

a) "Ligero.

b) Fácil de entender

c) Extremadamente difícil de llegar a dominar" (Schwaber y Sutherland, 2017)

Historia.

El marco de trabajo Scrum surge de un artículo publicado en 1986 por la Harvard Business Review titulado *The New Product Development Game* de Takeuchi y Nonaka, que introdujo las mejores prácticas más usadas en 10 empresas japonesas conocidas por su alta innovación. A partir de ese momento y tomando como referencia el juego de rugby, en 1993 Jeff Sutherland creó el proceso Scrum para ser utilizado en desarrollo de *software*; en 1995 Ken Shwaber realiza la primera publicación sobre Scrum, a partir de entonces ambos, juntos o separados han realizado numerosas publicaciones sobre el marco de trabajo Scrum (Rubin, 2012, p. 3).

Características de Scrum: Eventos, roles, herramientas, reglas.

Scrum define un proceso empírico, iterativo e incremental de desarrollo y bajo tres pilares propone su implementación:. transparencia, inspección y adaptación (Schwaber & Sutherland, 2017). Según Hundermark (2015), la esencia de Scrum es:

• El equipo recibe objetivos claros

• El equipo se organiza alrededor del trabajo

• El equipo entrega regularmente las características más valiosas

• El equipo recibe comentarios de personas fuera de sí mismo

• El equipo reflexiona sobre su forma de trabajar para mejorar

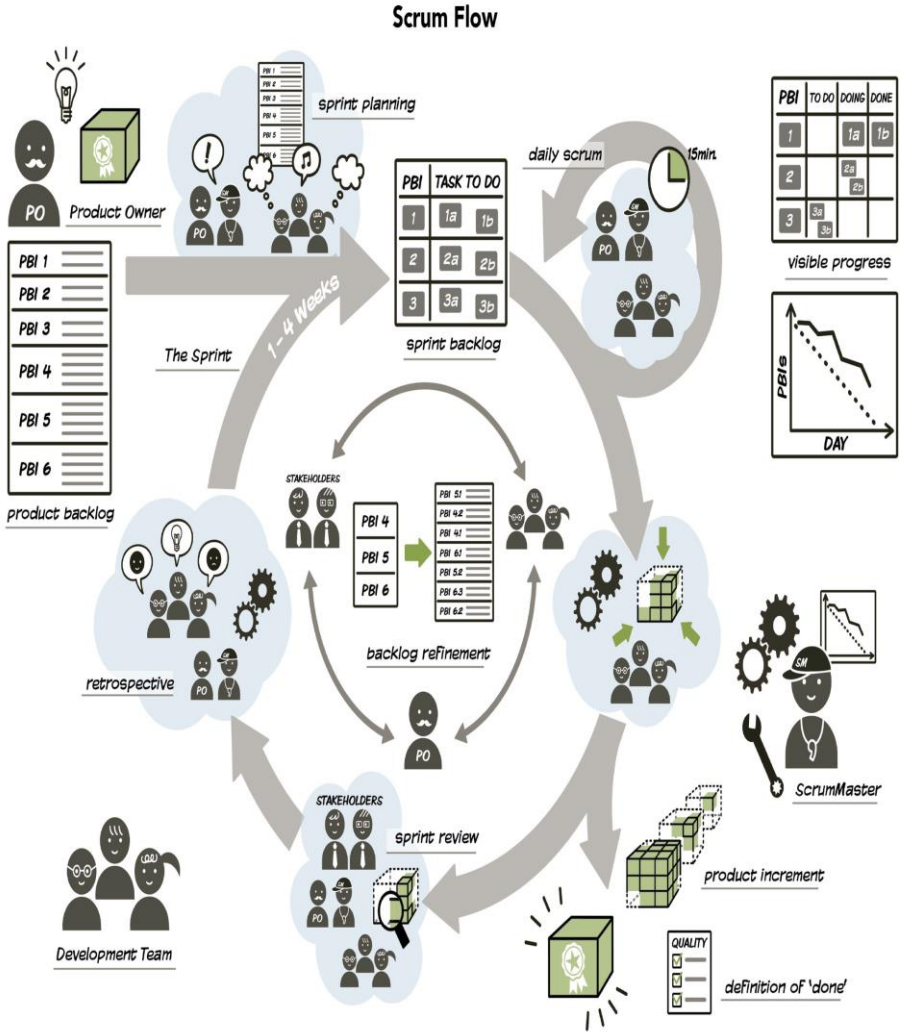
• Toda la organización tiene visibilidad del progreso del equipo

• El equipo y la administración comunican honestamente sobre el progreso y los riesgos. (Hundermark, 2015, p. 9)

A continuación, en la figura 5, se presenta el ciclo de eventos, plantillas y roles de los participantes del ciclo Scrum.

Figura N.º 5

Ciclo Scrum



Nota: Hundermark (2015).

El proceso parte de la lista de requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente ha priorizado los requisitos balanceando el valor que le aportan respecto a su costo y han sido divididos en iteraciones y entregas (Hundermark, 2015).

Eventos.

Sprint.

De acuerdo con Hundermark (2015), cuando se trabaja en Scrum, un proyecto se ejecuta en bloques temporales conocidos como iteraciones o *sprints* entre una y cuatro semanas de duración, según defina el equipo. La duración del *sprint* es fija y nunca se extiende una vez que este ha comenzado. El *sprint* es el ritmo de los ciclos de Scrum y está delimitado por la reunión de planificación del *sprint* y la reunión retrospectiva.

Según Hundermark (2015), el objetivo del *sprint* es lograr la meta planteada por el equipo al inicio del mismo.

Planeación del Sprint (Sprint Planning).

Hundermak (2015) describe el evento de planificación de *sprint* como la actividad que marca el comienzo de cada *sprin*t. Su propósito es planificar el trabajo que harán durante el mismo. Agrega que el evento se divide en dos partes, cada una con un propósito distinto, de la siguiente manera:

Parte 1.

La primera parte del Sprint Planning responde a la pregunta: "¿Qué podemos hacer y entregar al final de este Sprint? ". Entonces, a menudo, en la primera parte del evento se obtiene el "¿Qué?".

Comenzando desde la parte superior de su *product backlog*, el *product owner (*cliente) presenta el conjunto de características que le gustaría que el equipo desarrolle. El equipo de desarrollo hace preguntas para entender la necesidad con suficiente detalle, con el fin de que puedan pronosticar lo que pueden ofrecer durante el *sprint.* El equipo de desarrollo solo decide qué se puede lograr, teniendo en cuenta la duración del *sprint*, el tamaño y las capacidades actuales de sus miembros, la definición de hecho (Done), posibles interrupciones en su trabajo y esfuerzo requerido para desarrollar las necesidades. El *product owner* está presente en toda la parte 1 para dirigir el equipo de desarrollo y para responder preguntas. El Scrum máster asegura a cualquier otra parte interesada necesaria para ayudar al equipo de desarrollo a entender los requisitos.

Al final de la primera parte, el equipo de desarrollo se compromete con el *product owner* con aquello que creen pueden trabajar y finalizar en el presente *sprint* (Hundermark, 2015).

Estimación del esfuerzo requerido

Parte 2.

Según Hundermark (2015), la parte 2 del Sprint Planning responde a la pregunta: "¿Cómo haremos el trabajo?", dado esto se conoce la segunda parte del Sprint Planning como el evento "¿Cómo?".

En esta sesión, el equipo de desarrollo ayuda a crear un diseño de alto nivel de las características que acaba de comprometerse a entregar al *product owner*. Durante la segunda parte, el equipo puede tener preguntas adicionales con respecto a los requisitos, el Scrum máster debe garantizar que el *product owner* u otros interesados estén disponibles para responder. Durante el *sprint*, el equipo puede identificar nuevas tareas necesarias para completar cada requisito, en este caso, se agregan a cada requisito, a esto se le conoce como diseño emergente (Hundermark, 2015).

Daily Scrum Meeting.

Para Hundermark, el *Daily Scrum meeting* es uno de los principales puntos de inspección y adaptación en el ciclo de Scrum. El equipo de desarrollo se reúne para comunicar y sincronizar su trabajo y crear un plan para las próximas 24 horas, donde el equipo de desarrollo se autoorganiza constantemente para lograr el objetivo de *sprint*. En su libro *Do Better Scrum*, Hundermark expone que esta forma de colaboración entre el equipo es esencial para garantizar un progreso continuo y evitar bloqueos laborales. Además, explica que el evento diario de Scrum no es para informar el progreso, que la duración máxima recomendada es de 15 minutos y que es un evento propio del equipo de desarrollo y el Scrum Máster.

De acuerdo con los pioneros de Scrum Schwaber y Sutherland (2017), cada miembro del equipo debe responder las siguientes preguntas en un intervalo de tiempo de cómo máximo 15 minutos:

• ¿Qué se ha hecho desde la última reunión de sincronización? ¿Se pudo hacer todo lo que se tenía planeado? ¿Cuál fue el problema?

• ¿Qué se va hacer a partir de este momento?

• ¿Qué impedimentos se tienen para cumplir con los compromisos en esta iteración y en el proyecto?

Revisión del Sprint (Sprint Review).

Tal como lo indican Schwaber y Sutherland (2017) en su guía, el *Sprint Review* es una sesión que sirve para inspeccionar el incremento y adaptar el *backlog* del producto cada vez que sea necesario. Durante la revisión de *sprint*, el equipo Scrum muestra lo que trabajó durante el *sprint* y basándose en esto y en cualquier cambio al *backlog* durante el *sprint*, los asistentes colaboran para determinar los siguientes requerimientos que podrían hacerse con el fin de optimizar el valor.

Se trata de una reunión informal, no una reunión de seguimiento y la presentación del incremento tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración (Schwaber y Sutherland, 2017, p. 13). Para Schawaber y Sutherland (2017), el *Sprint review* consta de los siguientes elementos:

- Los asistentes son el Equipo Scrum y los interesados clave invitados por el Dueño de Producto.
- El *product owner* explica qué elementos del *backlog* se han "Terminado" y cuales no se han "Terminado".
- El Equipo de Desarrollo habla acerca de qué fue bien durante el Sprint, qué problemas aparecieron y cómo fueron resueltos esos problemas.
- El Equipo de Desarrollo demuestra el trabajo que ha "Terminado" y responde preguntas acerca del Incremento.
- El *product owner* habla acerca del *backlog* en el estado actual. Proyecta fechas de finalización probables en el tiempo basándose en el progreso obtenido hasta la fecha (si es necesario).
- El grupo completo colabora acerca de qué hacer a continuación, de modo que la Revisión del Sprint proporcione información de entrada valiosa para Reuniones de Planificación de Sprints subsiguientes. (Schwaber y Sutherland, 2017, p. 13)

Retrospectiva.

Conforme a Hundermark (2015), la retrospectiva de *sprint* es el evento final del *sprint*, se realiza después del Sprint Review y "se centra en el proceso: la forma en que el equipo de Scrum trabaja en conjunto, incluidas sus habilidades técnicas y las prácticas de desarrollo de *software* y las herramientas que están utilizando" (Hundermark, 2015, p. 25).

Para Schwaber y Sutherland (2017), el propósito de la retrospectiva de *sprint* es:

> Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas;
>
> Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras; y,
>
> Crear un plan para implementar las mejoras a la forma en la que el Equipo Scrum desempeña su trabajo. (Schwaber y Sutherland, 2017, p. 14)

Roles.

Según lo propuesto por Schwaber y Sutherland (2017) los equipos Scrum son:

a) "Auto organizados y multifuncionales.

b) Son capaces de llevar a cabo su trabajo sin depender de otros externos al equipo.

c) Diseñado para optimizar la flexibilidad, la creatividad y la productividad" (Schwaber y Sutherland, 2017).

Product owner.

Schwaber y Sutherland (2017) plantean en su guía que el *product owner:* "es el responsable de maximizar el valor del producto y del trabajo del Equipo de Desarrollo" . Dentro de sus principales funciones están:

a) Responsable de gestionar la pila de producto (Product *Backlog*).

b) Expresa claramente los elementos de la pila.

c) Ordena los elementos según prioridades.

d) Optimiza el valor del trabajo del Equipo de Desarrollo .

e) Asegura que la pila de producto sea transparente visible y clara, y muestra en lo que el trabajará a continuación.

f) Asegura que el Equipo de Desarrollo entiende los elementos de la pila. (Schwaber & Sutherland, 2017).

Scrum máster.

Según lo propuesto por Schwaber y Sutherland (2017), el Scrum máster: “es el responsable de asegurar que Scrum es entendido y adoptado. Es un líder que está al servicio del Equipo Scrum” .

Según la teoría propuesta en dicha guía, el Scrum máster ayuda a todos a entender cuáles iteraciones agregan valor al equipo, agrupándolas de la siguiente manera:

Servicio al *product owner*:

a) Encontrar técnicas para gestionar la lista de producto de manera efectiva.

b) Ayudar al equipo Scrum a entender la necesidad de contar con elementos de lista de producto claros y concisos.

c) Entender la planificación del producto en un entorno empírico.

d) Asegurar que el *product owner* conozca cómo ordenar la lista de producto para maximizar el valor.

e) Entender y practicar la agilidad.

f) Facilitar los eventos de Scrum según se requiera o necesite.

Servicio al Equipo de Desarrollo:

a) Guiar al Equipo de Desarrollo a ser auto organizado y multifuncional.

b) Ayudar al Equipo de Desarrollo a crear productos de alto valor.

c) Eliminar impedimentos para el progreso del Equipo de Desarrollo.

d) Facilitar los eventos de Scrum según se requiera o necesite.

e) Guiar al Equipo de Desarrollo en el entorno de organizaciones en las que Scrum aún no ha sido adoptado y entendido por completo.

Servicio a la organización:

a) Liderar y guiar a la organización en la adopción de Scrum.

b) Planificar las implementaciones de Scrum en la organización.

c) Ayudar a los empleados e interesados a entender y llevar a cabo Scrum y el desarrollo empírico de producto.

d) Motivar cambios que incrementen la productividad del equipo Scrum.

e) Trabajar con otros Scrum Masters para incrementar la efectividad de la aplicación de Scrum en la organización. (Schwaber & Sutherland, 2017)

Equipo de Desarrollo (Development team).

En la Guía de Scrum, Schwaber y Sutherland (2016) definen que el equipo de desarrollo realiza el trabajo de entregar un incremento de producto "Terminado" que potencialmente se pueda poner en producción al final de cada *sprint*, además, se indica que los equipos de desarrollo tienen las siguientes características:

- Son auto organizados. Nadie (ni siquiera el Scrum máster) debe indicar al Equipo de Desarrollo cómo convertir elementos de la lista del producto en incrementos de funcionalidad potencialmente desplegables;
- Los equipos de desarrollo son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un Incremento de producto;
- Scrum no reconoce títulos para los miembros de un Equipo de Desarrollo, todos son desarrolladores, independientemente del trabajo que realice cada persona; no hay excepciones a esta regla;
- Scrum no reconoce sub equipos en los equipos de desarrollo, no importan los dominios particulares que requieran tenerse en cuenta, como pruebas o análisis de negocio; no hay excepciones a esta regla; y,
- Los miembros individuales del Equipo de Desarrollo pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en el Equipo de Desarrollo como un todo. (Schwaber & Sutherland, 2017).

Tamaño del equipo de desarrollo.

El tamaño óptimo del equipo de desarrollo debe ser lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar una cantidad de trabajo significativa, se recomienda que el equipo no sobrepase las nueve personas (Schwaber & Sutherland, 2017).

Otros roles.

Para Hundermark (2015), en Scrum no hay un rol de administrador de proyectos, sino que las responsabilidades del administrador de proyectos tradicional se dividen en los tres roles en el equipo de Scrum, donde el *product owner* gestiona el producto (y el rendimiento de la inversión), el Scrum máster administra el proceso y el equipo de desarrollo se autogestiona.

Esto desde el punto de vista del autor Hundermark (2015) es un desafío para las personas que actualmente cumplen este rol y para las organizaciones que han trabajado con métodos tradicionales.

Plantillas o artefactos.

Lista de producto (product backlog).

De acuerdo con Schawaber y Sutherland (2017), la lista de producto es una lista ordenada de todo lo que se considera necesario para el producto, afirma que esta es la única fuente de requisitos por desarrollar y que el *product owner* es el responsable de asegurar su contenido, disponibilidad y ordenación.

Según Schawaber y Sutherland (2017), la lista de producto cambia constantemente y enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios que se deben realizar como resultado de la identificación de todo aquello que el producto necesita para ser de valor.

Se definen como atributos de la lista de producto: la descripción, la ordenación, la estimación y el valor (Schwaber & Sutherland, 2017).

Lista de pendientes del sprint (Sprint backlog).

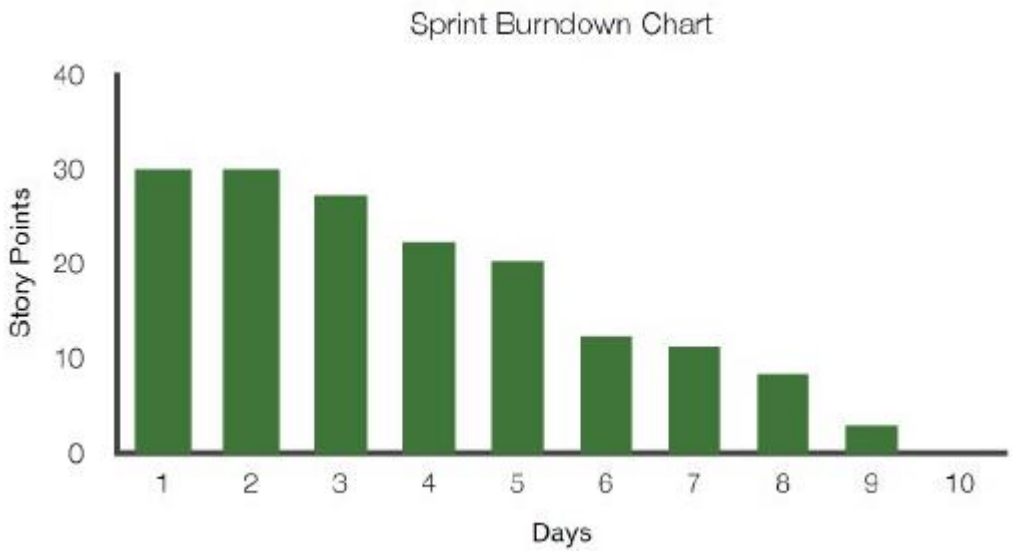
Schawaber y Sutherland (2017) definen la lista de pendientes del *sprint* como la lista de elementos seleccionados del listado de producto que deben ser desarrollados durante el *sprint*, incluye el plan necesario para lograr un "incremento terminado" al finalizar este.

Esta lista permite ver las tareas donde el equipo presenta inconvenientes y no avanza, lo que la convierte en una herramienta para la toma de decisiones. La lista contiene las tareas, el esfuerzo pendiente para finalizarlas y el responsable de cada tarea como resultado de la auto-asignación que han hecho los miembros del equipo. Para poder ver el progreso del *sprint* y su velocidad con respecto a tareas u horas pendientes, se utiliza un gráfico de trabajo pendiente (Gráfica Burndown) (Schwaber & Sutherland, 2017).

Hundermark (2015) define la gráfica de trabajo pendiente (Gráfico Burndown) como un indicador de si el equipo de desarrollo cumplirá con la meta del *sprint*, para esto propone el uso de los *story points* y la medición de su ejecución a lo largo del *sprint*.

Figura N.º 6

Burndown Chart



Nota: Hundermak (2015).

En esta gráfica se observa el inicio del *sprint* en día 1 con 30 *story points*, conforme avanza el *sprint,* la cantidad de puntos pendientes debe ir disminuyendo, de forma tal que se vaya visualizando el alcance de la meta conforme se acerca el fin del *sprint*,

Incremento.

Un Incremento es:

> La suma de todos los elementos de la Lista de Producto completados durante un *Sprint* y el valor de los incrementos de todos los *Sprint*s anteriores. Al final de un *Sprint*, el nuevo Incremento debe estar "Terminado", lo cual significa que está en condiciones de ser utilizado y que cumple la Definición de "Terminado" del Equipo Scrum. El incremento debe estar en condiciones de utilizarse sin importar si el *product owner* decide liberarlo o no. (Schwaber & Sutherland, 2017, p. 17)

Kanbam.

Según lo definido por Rasmusson (2010), Kanban es un sistema de señalización basado en tarjetas que Toyota desarrolló para coordinar el reabastecimiento de piezas en sus líneas de ensamblaje. Agrega que en Kanban el trabajo está limitado por un concepto llamado trabajo en progreso (WIP).

Según recomendaciones dadas por Rasmusson, Kanban es una buena práctica para operaciones y equipos de soporte que necesitan reaccionar rápidamente y no tienen la definición de iteraciones de duración fija, sin embargo, si la madurez del equipo es baja, aconseja mantener iteraciones de longitud fija.

Por otro lado, para Rubin (2012), Kanban no es una solución de proceso independiente, sino un enfoque que se superpone a un proceso existente, asegura que Kanban defiende la visualización de cómo fluye el trabajo a través del sistema, limitando el trabajo en proceso (WIP) en cada paso, para asegurarse de que el equipo no esté haciendo más trabajo del que puede hacer.

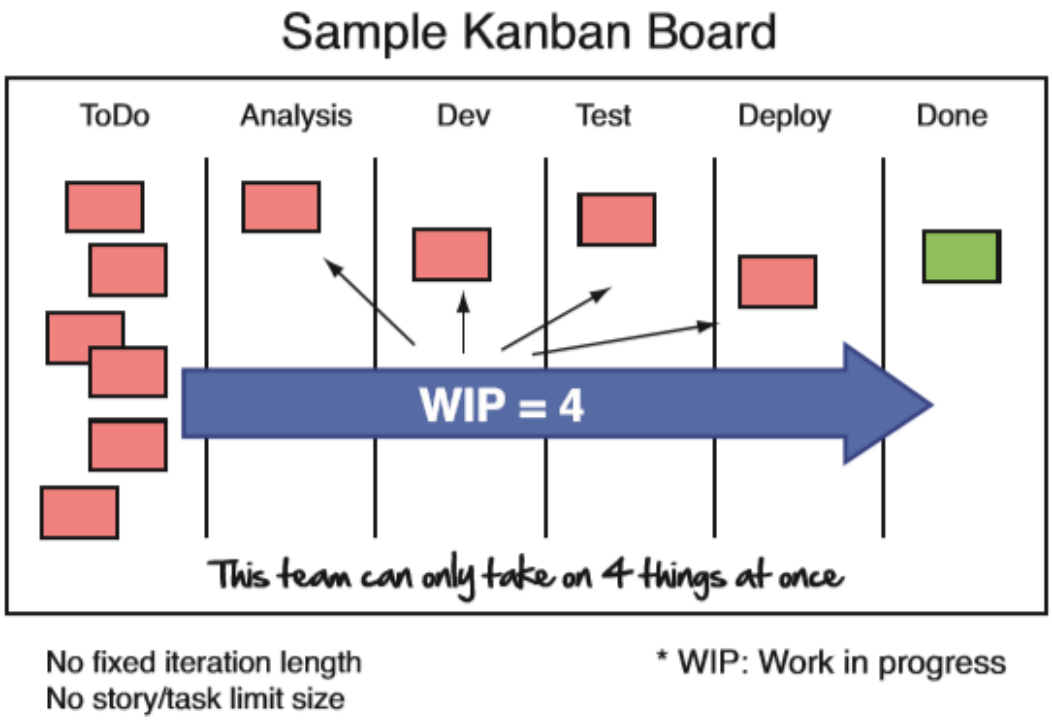
Rubin basado en el criterio de otros expertos que han utilizado Kanban señala que el enfoque de Kanban en la eliminación de la sobrecarga (alineando el WIP con la capacidad) y la reducción de la variabilidad en el flujo y la forma en que fomenta un enfoque evolutivo para el cambio, hace que sea apropiado su uso en dominios simples o complejos.

Scrum y Kanban, ambos son enfoques ágiles para el desarrollo y cada uno tiene fortalezas y debilidades que deben considerarse en el momento en que se conozca el detalle del proyecto. Para Rubin (2012), ambas pueden usarse para abordar las diferentes necesidades del sistema, de

tal manera que Scrum se puede usar para el desarrollo de nuevos productos y Kanban para soporte y mantenimiento de los sistemas.

Figura N.º 7

Pizarra Kanban



Nota: Rasmusson (2010).

Tal como muestra la figura y según lo que explica Rasmusson (2010), en la pizarra de Kanban se divide el trabajo en bloques, se utilizan columnas con nombre para ilustrar el estado en que está cada elemento en el flujo de trabajo, se escribe cada elemento en una tarjeta y se coloca en la columna correspondiente según su estado. Conforme se va avanzando, las tarjetas se van moviendo de columna.

Scrumban.

Para Swisher (2014), Scrumban representa los mejores elementos de Scrum y Kanban, donde se unen conceptos de un equipo trabajando en conjunto para completar el trabajo y la cantidad de trabajo limitado como un conjunto de prácticas dentro de una metodología para lograr altos rendimientos y obtener siempre visibilidad del desarrollo del proceso.

De acuerdo con Swisher (2014), los siguientes elementos o artefactos son claves en Scrumban:

- Lista de trabajo (*Work Backlog*): similar a la lista de producto en Scrum (*product backlog*) donde el *product owner* identifica, prioriza, aclara y da mantenimiento a las historias de usuario y define los criterios de aceptación respectivos que agreguen valor al negocio.
- Ítem de trabajo (*Work ítem*): descripciones de las funcionalidades solicitadas por el cliente ya sean mejoras o defectos.
- Línea de trabajo (*Work line*): etapa del proceso de desarrollo que jala el trabajo desde la lista de trabajo hasta la entrega de un producto de valor al cliente.
- Etapas: pasos requeridos para desarrollar *software*.

Límites del trabajo en proceso (WIP Limits).

Para Swisher (2014), los límites del trabajo en proceso (WIP) se diseñan para mantener un equilibrio de la carga de trabajo que maneja el equipo, este equilibrio está basado en lograr evidenciar los óptimos niveles de cargas para cada etapa de la línea de trabajo. Siguiendo la recomendación de Swisher (2014), una de las formas en que se puede definir el WIP inicial es dividiendo el total de recursos entre dos, si el resultado es un número impar, se redondea hacia arriba.

Tarjetas Kanban.

Las tarjetas Kanban representan los ítems de trabajo que forman parte de la lista de trabajo, la información que contiene cada tarjeta puede ser el título y un identificador, o bien, completarse con parte de la historia de usuario, priorización y estadísticas de su avance (esta información no sustituye la historia de usuario completa para el desarrollador). Las tarjetas tienen el objetivo de funcionar como una guía del progreso para los miembros del equipo y reportes de estado para el *product owner* (Swisher, 2014).

Figura N.º 8

Tarjetas Kanban

B#042 Standard	Create Remember-Me Login Option	
As a customer, I want an option for the website to remember my login information so that I don't have to type it in each time.		
Workline Transit Statistics (in | out)		
Design	2013-08-01	2013-08-03
Coding	2013-08-03	2013-08-06
Testing	2013-08-07	2013-08-09

Nota: Swisher (2014).

Eventos Scrumban.

Swisher (2014) define dos eventos o ceremonias dentro del marco Scrumban:

Sesión diaria (DSU).

La sesión diaria tiene el mismo objetivo de la metodología Scrum, coordinar actividades e identificar inconvenientes, la duración recomendada es de 30 minutos, donde los primeros 15 minutos cada miembro del equipo comunicará lo que hizo desde la última sesión y lo que realizará el día en curso< los otros 15 minutos se utilizan para discutir temas necesarios para coordinar más actividades y revisar impedimentos que dificultan continuar con el trabajo (Swisher, 2014).

Sesión semanal (WTB).

Uno de los objetivos de la sesión semanal es mostrar el trabajo realizado y lograr la aprobación del *product owner*. Una vez mostrado el trabajo, el equipo explica el estado del proceso de desarrollo, de sus herramientas, de la lista de trabajo y de los elementos de trabajo a través de la línea de este. Se recomienda una duración de 2 a 4 horas y que la misma sea realizada a mitad de semana.

Roles.

Swisher (2014) menciona como roles el equipo Scrumban y el *product owner*, con las mismas funcionalidades identificadas en la metodología Scrum.

Extreme programming.

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito de los desarrollos de *software*, donde se promueve el trabajo en equipo, existe la preocupación por el aprendizaje de los desarrolladores y se propicia un buen clima de trabajo. XP se basa en una retroalimentación continua entre el cliente y el equipo de desarrollo, una comunicación fluida entre todos los participantes y una simplicidad en las soluciones implementadas. Promueve una actitud de valentía para enfrentar los cambios. XP se considera adecuada para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico (Wells, 2013).

Para Shore y Warden (2008), los equipos de XP trabajan enfatizando la colaboración cara a cara, realizan casi todas las actividades tradicionales de desarrollo de *software,* tales como el análisis, el diseño, la codificación, las pruebas e incluso el despliegue, con la diferencia de que su realización es con una frecuencia rápida y continua. Su planteamiento expone la ejecución de iteraciones con incrementos de trabajo semanales, donde cada semana el equipo hace un poco de planificación de lanzamiento, un poco de diseño, un poco de codificación y un poco de prueba.

Según Shore y Warden, se trabaja con historias que contienen características muy pequeñas o partes de características que tienen valor para el cliente. Semanalmente, el equipo se compromete a entregar de cuatro a diez historias, se trabaja en todas las fases de desarrollo para cada historia. Al final de la semana, implementan el *software* trabajado para su revisión interna.

De acuerdo con Shore & Warden (2008), las etapas en Xtreme Programming se definen de la siguiente manera:

Planificación.

Cada equipo de XP incluye varios expertos en negocios, responsables de tomar decisiones comerciales que apunten al proyecto en la dirección correcta, aclaran la visión del proyecto, crean historias, elaboran un plan de lanzamiento y gestionan los riesgos. Los programadores proporcionan estimaciones y sugerencias, que se combinan con las prioridades del cliente en un

proceso llamado juego de planificación. Juntos el equipo se esfuerza por crear lanzamientos pequeños y frecuentes que maximicen el valor.

> El esfuerzo de planificación es más intenso durante las primeras semanas del proyecto. Durante el resto del proyecto, los clientes continúan revisando y mejorando la visión y el plan de lanzamiento para dar cuenta de las nuevas oportunidades y eventos inesperados. Además del plan de lanzamiento general, el equipo crea un plan detallado para la próxima semana al comienzo de cada iteración.
>
> El equipo toca base todos los días en una breve reunión de pie, y su espacio de trabajo informativo mantiene a todos informados sobre el estado del proyecto. (Shore & Warden, 2008, p. 38)

Análisis.

En la metodología XP, en lugar de utilizar una fase de análisis inicial para definir los requisitos, los clientes se mantienen disponibles para el equipo durante todo el proyecto. Para definir los requerimientos, los clientes utilizan su propio conocimiento como clientes combinados con técnicas tradicionales de recopilación de requisito. Los clientes son responsables de organizar su trabajo para que estén listos cuando los programadores soliciten información, identifican los requisitos generales de una historia antes de que los programadores la estimen y los requisitos detallados antes de que los programadores los desarrollen (Shore & Warden, 2008).

Diseño y codificación.

Shore y Warden (2008) definen el diseño y la codificación como la unión de prácticas de desarrollo y *testing* entre las que destacan el diseño incremental y arquitectura para crear y mejorar continuamente el diseño en pequeños pasos, ejecución de pruebas (TDD), *peer review*, control de versiones para la gestión de configuración y mantenimiento de su propia compilación automatizada, proponen la integración del código continua .Para respaldar este esfuerzo, también proponen el mantenimiento de los estándares de codificación.

Testing.

Para Shore y Warden (2008), cada miembro del equipo -programadores, clientes y evaluadores- hacen su propia contribución a la calidad del *software* de la siguiente manera:

- Los programadores proporcionan la primera línea de defensa con desarrollo basado en pruebas TDD, produciendo unidades automatizadas y pruebas de integración. Estas pruebas ayudan a garantizar que el *software* haga lo que los programadores se propusieron.

- Los clientes al realizar pruebas ayudan a garantizar que la intención de los programadores coincida con sus expectativas, revisan el trabajo en progreso para garantizar que la aplicación funcione de la manera que esperan.

Cada vez que los programadores integran su código, ejecutan todo el conjunto de pruebas de regresión para comprobar si existen conflictos. El equipo también apoya sus esfuerzos de calidad a través de la programación de pares, el trabajo energizado y la holgura de iteración (Shore & Warden, 2008).

Deployment.

Shore y Warden manifiestan que los equipos de Deployment XP mantienen su *software* listo para implementar al final de cualquier iteración. Implementan el *software* todas las semanas como preparación para demostrar la iteración semanal a los interesados. El despliegue a clientes reales es programado de acuerdo con las necesidades del negocio.

De manera similar, cuando el proyecto finaliza, el equipo puede transferir las tareas de mantenimiento a otro equipo. En este caso, el equipo crea documentación y proporciona capacitación según sea necesario durante sus últimas semanas (Shore & Warden, 2008).

Roles XP.

Tal como lo describen Shore y Warden (2008), los equipos XP se sientan juntos en un espacio de trabajo abierto. Al comienzo de cada iteración, el equipo se reúne para una serie de actividades: una demostración de iteración, una retrospectiva y una planificación de iteración. Estos generalmente toman de dos a cuatro horas en total. El equipo también se reúne para las reuniones diarias de pie, que generalmente toman de cinco a diez minutos cada una.

Además de estas actividades programadas, exponen que todos en el equipo planifican su propio trabajo, sin embargo, explican que eso no significa que todos trabajen de forma independiente; simplemente no están en un horario explícito. Los miembros del equipo determinan los detalles de cada reunión cuando lo necesitan. A veces es tan informal como que alguien se ponga de pie y anuncie a través del espacio de trabajo compartido que le gustaría

discutir un tema. Esta auto organización es un sello distintivo de los equipos ágiles (Shore & Warden, 2008).

Para Shore & Warden (2008), los roles que deben existir son los siguientes: product manager, expertos, diseñadores de la interacción, analistas de negocio, programadores, diseñadores y arquitectos, especialistas técnicos y *testers.*

Design Thinking.

Tal como establece IBM (2016), al lograr tener empatía con el usuario, los diseñadores pueden trabajar hacia los resultados que satisfagan las necesidades de los mismos con más éxito. Este enfoque centrado en el usuario se conoce como Design Thinking, mismo que permite a los diseñadores y otros profesionales abordar una amplia gama de negocios complejos y problemas sociales.

IBM (2016) define *LOOP* como un modelo de comportamiento para comprender las necesidades de los usuarios, lograr entregables continuos y prever un futuro mejor basado en un ciclo continuo de observación, reflexión y creación. Las claves para lograr el éxito de la implementación de la metodología Design Thinking son las siguientes:

1- *Hills* (colinas): se deben alinear a los equipos complejos en torno a un entendimiento común de los resultados esperados y de esta manera, se convierten las necesidades de los usuarios en metas del proyecto. Las colinas están compuestas por un "Quién" (un usuario específico o grupo de usuarios), un "Qué" (una acción o habilitación específica) y un "Wow" (un diferenciador medible del mercado). Se deben escoger solamente tres colinas, el resto de las colinas pueden ser abordadas en futuras versiones.
2- *Playbacks*: consiste en lograr reunir al equipo y a los interesados en un espacio seguro e inclusivo para reflexionar sobre el trabajo, para así obtener consenso y compartir el trabajo en progreso en el camino, se muestra cómo la herramienta o concepto resuelve un problema en el trabajo del mundo real de su usuario.
3- Usuarios del patrocinador: contar con la colaboración de usuarios reales para aumentar la velocidad y cerrar la brecha entre sus suposiciones y la realidad de los usuarios. Los usuarios del patrocinador son usuarios o usuarios potenciales que tienen y comparten su experiencia con el equipo, son participantes activos que trabajan con el equipo para ayudarlo a entregar un resultado que satisfaga sus necesidades. El objetivo es diseñar para

usuarios meta reales, en lugar de las necesidades imaginadas. Los usuarios del patrocinador deben ser personas reales, no personas o "tipos", participan con el equipo durante todo el desarrollo y asisten a los *playbacks* (IBM, 2016).

Herramientas colaborativas

Gestión de proyectos

Jira.

Jira *software* es una herramienta de gestión de proyectos e incidencias para equipos ágiles, el cual ofrece las siguientes funcionalidades: tableros Scrum, tableros Kanban, *agile reporting*, planificación de hojas de ruta ágiles, planifica hojas de ruta visuales, pronóstico de las fechas de publicación, gestiona los recursos, analiza varias situaciones, supervisa los objetivos de la empresa, crea informes del estado y el progreso (Atlassian, 2017).

Team Foundation Server.

Team Foundation Server ofrece funciones de control de código fuente, seguimiento de elementos de trabajo, Team Foundation Build, un sitio web del portal del proyecto de equipo, creación de informes y administración de proyectos. Incluye, además, un almacén de datos donde se guardan los datos de seguimiento de elementos de trabajo, el control de código fuente, las compilaciones y las herramientas de pruebas (Microsoft, 2017).

Gestión de la calidad

Practitest.

Según Montvelisky (2017), Practitest es una herramienta de gestión de control de calidad profesional para las pruebas manuales y de automatización organizándolas según ciclos, *sprint*s u otro tipo de iteración utilizada. Logra integrar sin problemas sus pruebas manuales con sus procesos de automatización, reutilizarlas y correlacionar los resultados entre diferentes lanzamientos y productos, de esta manera garantiza poder liberar Sus productos con confianza y control.

Además, garantiza la visibilidad del proyecto con la ayuda de excelentes paneles de control e informes profesionales, generando automáticamente información relevante para la

gerencia y el resto de la organización, así como la creación de cuadros de mando e informes profesionales para compartir con todas las partes interesadas (Montvelisky, 2017).

Zephir.

Zephyr para JIRA es una aplicación complementaria que aumenta JIRA, brindando capacidades de administración de pruebas rentables y altamente sofisticadas dentro de JIRA. Juntos, Zephyr para JIRA y JIRA Cloud y Sever, permiten a los desarrolladores, probadores y al equipo de proyecto completo estar preparados en cada etapa del ciclo de vida del *software*, para planificar, desarrollar pruebas y lanzar un excelente *software* (Zephyr, 2016).

Release management

Bitbucket.

Bitbucket es un sistema de control de versiones distribuidas que facilita la colaboración con su equipo, logrando la aprobación de la revisión del código de manera más eficiente con las solicitudes de extracción, manteniendo discusiones directamente en el código fuente con comentarios en línea. Además, es considerado un modelo de implementación flexible para equipos de todos los tamaños y necesidades, además, cuenta con la facilidad de integración con otras herramientas tales como Jira, Hipchat, Bamboo, Trello (Bitbucket, 2017).

Github.

La herramienta Github permite realizar compromisos inteligentes agregando comentarios, tiempo y problemas de transición, además, es de fácil instalación, resume y ordena los datos de confirmación y permite al usuario recibir notificaciones por correo electrónico cuando vea un problema o repositorio, así como ver la lista de eventos y actividades del usuario actual y otras personas que trabajan en el proyecto (GitHub, 2017).

Blue Mix con Github.

IBM® Cloud Dedicated para GitHub Enterprise es la versión administrada y alojada en la nube de IBM de GitHub Enterprise, disponible para entornos dedicados de IBM Cloud. GitHub proporciona la experiencia de codificación social, mientras que IBM Cloud Dedicated proporciona un entorno de computación en la nube en *hardware* físicamente aislado que está integrado en su red.

Git es un sistema distribuido de control de versiones. Con Git, se puede hacer un seguimiento de los cambios de la aplicación y colaborar con el equipo de una manera ágil. Git es rápido porque sus repositorios se almacenan localmente en lugar de en una computadora remota.

Cuando los equipos usan Git correctamente, la rama maestra siempre está lista para implementarse. De esta forma, los equipos pueden entregar con frecuencia y practicar tanto la integración continua como la entrega continua. Además, GitHub se integra con IBM Cloud para admitir el seguimiento de cambios y lograr una implementación automatizada (IBM, 2017).

Ecosistema IBM

IBM como tal cuenta con un ecosistema que cubre la gestión de requerimientos, gestión de desarrollos y *testing* especializado (Doors, Rational Team Concert, Rational Funtional testing, IBM Performance tester, IBM Integration tester, AppScan IBM security) (IBM, Blue Mix Devops, 2017).

Ingeniería de requerimientos

Historias de usuario.

Las historias de usuario son utilizadas para especificar los requisitos del *software*, para Rasmusson (2010), uno de los elementos más importantes de las historias de usuario es que su contenido sea valioso, es decir, que sea algo por lo cual el cliente pagaría. Deben estar escritas en términos simples de fácil comprensión por usuarios del negocio, evitando el uso de información técnica.

Entre las principales características de una buena historia de usuario están: independiente, de forma tal que en la medida de lo posible si se cambia una historia de usuario no afecte otras; negociable, se deben poder negociar, si es necesario, características de la historia de usuario según lo que se puede pagar o según las restricciones que se presenten; testeable, de modo que con su información se puedan redactar casos de prueba que permitan asegurar que lo trabajado funciona tal y como se solicitó; pequeña y estimable, para poder desarrollarlas dentro de las iteraciones y poder estimar de una forma más confiable (Rasmusson, 2010).

Buenas prácticas de *software*

Peer review.

Para Cohen, Teleki y Brown (2013), Peer review consiste en evaluar y garantizar la calidad del código fuente que se desarrolla en el equipo, así como mantener la integridad del código, se define el Peer review como un elemento lógico en el proceso de desarrollo. Se anotan los siguientes beneficios al utilizar Peer review:

-Mejora de la calidad del código.

-Disminución de defectos en el código.

-Mejora en la comunicación del contenido del código.

-Educación para programadores principiantes.

Además, se identifican los siguientes beneficios indirectos:

-Disminución del tiempo de desarrollo y ciclos de pruebas.

-Menor necesidad de soporte técnico.

-Mayor satisfacción del cliente.

-Código más amigable para dar mantenimiento (Cohen, Teleki, & Brown, 2013).

Test driven development TDD.

Según Kniberg (2007), el desarrollo basado en pruebas significa que se escribe una prueba automatizada, luego se desarrolla el código suficiente para hacer que pase esa prueba, después, refactoriza el código principalmente para mejorar la legibilidad y eliminar la duplicación, es decir, primero se piensa en la prueba y después se define el código, para asegurar que pase la prueba, asegura que este proceso se continúa las veces necesarias hasta finalizar el proyecto.

Unit testing.

Según lo expuesto por Osherove (2014), las pruebas unitarias no son un concepto nuevo en el desarrollo de *software* y desde la década de 1970, se demuestra una y otra vez que consisten en una de las mejores formas en que un desarrollador puede mejorar la calidad del código

mientras que obtiene una comprensión más profunda de los requisitos funcionales de una clase o método.

Se define la prueba unitaria como la prueba de una parte de un código (generalmente un método) que invoca otra pieza de código y verifica la corrección de algunas suposiciones después. Si las suposiciones resultan ser incorrectas, la prueba de la unidad ha fallado. Se define una unidad como un método o función (Osherove, 2014).

CAPÍTULO III. MARCO METODOLÓGICO

Todo proceso de investigación necesita establecer los criterios metodológicos con base en los que se realiza, por lo tanto, en el presente capítulo se describe el enfoque, el diseño de la investigación, la población y muestra, las variables identificadas, así como sus indicadores y los instrumentos por utilizar para la recolección de los datos que determinan la confiabilidad y validez durante todo el proceso de investigación.

Enfoque

El enfoque cualitativo se guía por áreas o temas significativos de investigación. Los estudios cualitativos pueden desarrollar preguntas antes, durante o después de la recolección de datos y el análisis. Con frecuencia estas actividades sirven, primero para descubrir cuáles son las preguntas de investigación más importantes y después, para perfeccionarlas y responderlas. La acción indagatoria es dinámica en ambos sentidos: entre los hechos y su interpretación, y resulta un proceso más circular en el que la secuencia no siempre es la misma, pues varía con cada estudio (Hernández et al., 2014, p.7).

El enfoque cuantitativo representa un conjunto de procesos, es secuencial y probatorio, por lo que se no puede eludir ninguno de sus pasos. El orden es riguroso, pero sí se puede redefinir alguna fase. Parte de una idea que va acotándose y, una vez delimitada, se derivan objetivos y preguntas de investigación, se revisa la bibliografía y se construye un marco o una perspectiva teórica. De las preguntas se establecen hipótesis y se determinan variables. Posteriormente, se traza un plan para probarlas (diseño) y se miden las variables en un determinado contexto. Las mediciones obtenidas se analizan utilizando métodos estadísticos y se extrae una serie de conclusiones en relación con las hipótesis (Hernández et al., 2014, p.4).

Hernández et al. (2014, p. 534) resumen el enfoque mixto como aquel que utiliza evidencia de datos numéricos, verbales, textuales, visuales, simbólicos y de otras clases, para entender problemas en las ciencias. Es un enfoque que recolecta, analiza y vincula datos cualitativos y cuantitativos en un mismo estudio para responder a un planteamiento del problema, ayuda a clarificar y a formular el planteamiento del problema, así como las formas más apropiadas para estudiar y teorizar los problemas de investigación, así mismo propone la realización de una investigación de manera sistemática, profunda y con la calidad que se requiere, de forma tal que con frecuencia se recolecten datos cuantitativos y cualitativos, utilizando

elementos de ambos enfoques para alcanzar los objetivos y lograr responder las preguntas de la presente investigación (Hernández, 2017).

La presente investigación utilizará el enfoque mixto siguiendo las siguientes fases mostradas en la figura 9.

Figura N.º 9

Fases enfoque mixto (cualitativo integral)

Nota: elaboración propia.

En la primera fase se obtendrá la información previa y actual sobre la situación donde se realiza el estudio, de forma completa, dando prioridad a toda aquella información que esté vinculada con la necesidad de innovación planteada en este proyecto. Seguidamente, una vez se entienda el contexto, se determina cuáles son los datos que se deben recolectar y el tamaño de la muestra necesario.

Una vez definido lo anteriormente mencionado, se procede con la selección y creación de instrumentos, para continuar con la aplicación de los mismos y lograr así la recolección de todos los datos requeridos. Cuando se tiene la información recolectada, es necesaria la organización de los datos en una base que permita el análisis de los mismos y, posteriormente, proceder con este análisis que permite clasificar la información y descubrir patrones que lleven a resolver la necesidad de innovación planteada (Hernández, 2017).

Diseño (alcance)

Para Hernández et al. (2014), el diseño o alcance de una investigación puede ser de diferentes tipos: exploratorio, descriptivo, correlacional o explicativo. La presente investigación centrará su diseño en los siguientes tipos:

Investigación descriptiva

Busca especificar propiedades y características importantes de cualquier fenómeno que se analice. Describe tendencias de un grupo o población (Hernández et al., 2014, p. 92).

Investigación explicativa

Busca describir el fenómeno de forma tal que se obtenga la explicación del comportamiento de las variables tratando de lograr el descubrimiento de las causas (Hernández, 2017).

Para la presente investigación, se propone un estudio descriptivo- explicativo que logre descubrir de qué manera se puede implementar una metodología ágil en una compañía centrada en desarrollo por procesos y donde actualmente no existe una metodología estandarizada de desarrollo de *software* diferente a la tradicional, por tanto, será necesario realizar una investigación para obtener datos sobre la situación actual con respecto al desarrollo de *software* y así poder formular una estrategia adecuada con los datos recolectados.

Muestra de la investigación

Según lo expuesto por Hernández (2017), una muestra es un subgrupo de la población de interés donde se recolectan los datos necesarios para los análisis correspondientes que llevarán a hacia el cumplimiento de los objetivos.

Para seleccionar la muestra, Hernández (2017) propone dos estrategias: la probabilística y la no probabilística, sentando su principal diferencia en que, en las muestras probabilísticas, todas las unidades, casos o elementos de la población tienen al inicio la misma probabilidad de ser escogidos, se trabaja mediante procedimientos previamente establecidos, la estrategia no probabilística la elección de las unidades no depende de la probabilidad, sino de razones relacionadas con las características de la investigación.

La muestra de la presente investigación es de tipo no probabilístico, la cual está conformada por toda la población del área de Integration Solutions Engine de GBM, siendo alrededor de 40 profesionales en ingeniería de *software* e ingeniería industrial.

Variables

Según lo expuesto por Hernández (2017), una variable es una característica que puede fluctuar y cuya variación es susceptible a medirse u observarse. A cada variable se le puede dar una definición conceptual, que se refiere a la definición teórica o constitutiva, así como una definición operacional que indica que actividades u operaciones hay que realizar para medir una variable

Para la presente investigación, se identifican las siguientes características que por su naturaleza debe investigarse, observarse, y evaluarse. En la siguiente tabla 1 se detalla cada una de las mismas.

Tabla N.º 1

Descripción de variables: Conceptual, operativo e instrumental

Problema	Objetivo General	Objetivos específicos	Variables/Conceptual	Criterios de medición (Indicadores)	Instrumento (herramienta para recolectar los datos)
¿Cuáles metodologías ágiles y cuáles herramientas colaborativas se deben implementar para lograr una gestión donde las estimaciones de los proyectos y la toma de decisiones sucedan bajo menores grados de incertidumbre, con un mayor nivel de calidad	Diseñar una guía metodológica basada en prácticas ágiles adaptables a herramientas colaborativas para la gestión y ejecución de proyectos de *software* que permitan a la organización la innovación y consolidación como líderes en el mercado de desarrollo de *software*.	Investigar marcos de trabajo ágiles, así como herramientas colaborativas que permitan la gestión de proyectos de *software.*	Marcos de trabajos ágiles: Los métodos ágiles son procesos que soportan la filosofía ágil, consisten en elementos individuales llamados prácticas. Las prácticas incluyen el uso de control de versiones, el establecimiento de estándares de codificación y la entrega de demostraciones semanales a los interesados (Shore & Warden, 2008). Herramientas colaborativas: Herramientas de entrega continua que dan soporte a las tareas de desarrollo, pruebas, despliegue y de operaciones (IBM, Blue Mix Devops, 2017).	<u>Metodologías</u> Cantidad y tipo de roles. Cantidad y tipo de ceremonias. Métodos para medición de la productividad, avance y cumplimiento del alcance. <u>Herramientas</u> Costo Interfaz amigable Seguridad	Registro de tabla comparativa en Excel que permita la selección de las opciones ideales para el área en estudio.

y un desgaste menor de los equipos de trabajo, considerando la participación continua de los clientes para una mayor satisfacción de sus expectativas?					
		Determinar la propuesta idónea de marcos ágiles y herramientas colaborativas según las características actuales de la organización.	Características de la organización: Grupo de elementos que permiten analizar el estado actual de la organización tomando en cuenta debilidades, fortalezas, necesidades de la misma, tales como el logro de la satisfacción de sus clientes, tiempos de entrega y cumplimiento de objetivos (Lamb, Hair, & McDaniel, 2011).	-Debilidad interna de la organización. -Solidez interna de la organización. -Causa principal de la necesidad de innovación. -Factores prioritarios que limitan el proceso de innovación.	Guía conversatorio para líderes técnicos, Excel. Registro Diagrama de Ishikawa Excel. Entrevista tabla multivoto, Word. Entrevistas matriz EFI en Excel.
		Diseñar los procesos principales, métricas y artefactos de trabajo mínimos necesarios para poner en marcha la metodología propuesta.	Procesos: Un método o proceso es una forma de trabajar. Métricas: Una métrica es un tipo de medida que se utiliza para determinar un valor cuantificable dentro de una práctica o proceso (Heizer & Render, 2009). Artefactos: Un subproducto tangible producido durante el desarrollo del	Salidas y entradas de los procesos. Productividad de los equipos. Insumos de los procesos necesarios.	-Registro Diagrama de flujo, blueworks. - Registro Tabla de métricas en Excel -Registro de plantillas con la información requerida por artefacto en Word o

			producto. La acumulación de productos, la acumulación de datos acumulados y el incremento del producto potencialmente realizable son ejemplos de artefactos de Scrum (Rubin, 2012).		Excel.
		Documentar un plan de implementación de la propuesta como guía para la organización.	Plan de trabajo: planificación y gestión que permite llevar a cabo los fines de la organización, mediante una adecuada definición de los objetivos y metas que se pretenden alcanzar, de manera que se utilicen con eficiencia, eficacia y economicidad (Heizer & Render, 2009).	Fases Tareas Duración Trabajo Recursos Dependencias	Microsoft Project

Nota: elaboración propia.

Instrumentos

Para este proyecto, se utilizarán como instrumentos de recopilación de datos el registro del resultado de investigaciones y conversatorios con los equipos, los cuales serán utilizados con los recursos que conforman el área de *software* de la división de Integration de GBM.

Las preguntas que se van a realizar como parte de los conversatorios van a ser de tipo abiertas, ya que se quiere saber el estado actual del proceso de desarrollo de *software* y qué dificultades se presentan en este proceso, adicionalmente, se quiere identificar el conocimiento que se tiene de la metodología ágil, esto con el fin de conocer el nivel de madurez actual y las posibles dificultades que se pueden presentar al implementar la metodología.

Los registros se realizarán basados en el proceso de investigación previo según la necesidad de cada caso. Para ambos entrevistas y registros, se requiere de los siguientes recursos informáticos: Microsoft office, blueworks.

Tabla N.º 2

Descripción de instrumentos

Indicador	Instrumento	Recursos requeridos	Beneficios esperados
Características de cada herramienta	Registro tabla comparativa de herramientas colaborativas	Humanos e informáticos	Tener la información suficiente para poder comparar y seleccionar aquellas que mejor se adapten a GBM.
Nivel de madurez de los equipos de desarrollo de GBM	Cuestionario criterios de madurez prácticas de desarrollo	Humanos e informáticos	Conocer la madurez actual de los equipos para poder definir la barda de la metodología inicial, sus artefactos y métricas
Debilidad y solidez interna de la organización	Tabla de fortalezas y debilidades encontradas en el conversatorio con el personal	Humanos e informáticos	Conocer el grado de debilidad y solidez de la organización para poder seleccionar la metodología ágil que pueda ser adoptada con mayor facilidad.

Indicador	Instrumento	Recursos requeridos	Beneficios esperados
Identificar las características asociadas a los factores de la calidad	Causas identificadas en los conversatorios del porqué se está requiriendo un cambio metodológico y procedimental- Diagrama Ishikawa	Humanos e informáticos	Conocer todas las características asociadas a metodologías ágiles, tecnología, capital, etc. necesarias para su implementación.
Procesos	Registros Diagrama de flujo	Humanos e informáticos	Conocer la forma en que hoy en día se trabaja.
Productividad de los equipos	Registro Tabla de métricas	Humanos e informáticos	Identificar las métricas necesarias para medir la productividad de los equipos
Documentación requerida en ágil	Registro información requerida para artefactos	Humanos e informáticos	Identificar la información necesaria para hacer de una plantilla un instrumento de utilidad

Nota: elaboración propia.

Proceso para la recolección de datos

Para la recolección de los datos, es necesario aplicar algunas técnicas que a través de instrumentos permitan recabar la información necesaria para determinar las características y requerimientos de la metodología por diseñar. El método que se utilizará para la recolección de datos está basado en registros y conversatorios como herramientas principales, los cuales serán contestados por todo el personal del área de Integration.

Las bases de datos se elaborarán en programas de Microsoft office como Word, Excel, además de herramientas IBM como Blueworks.

Método de análisis

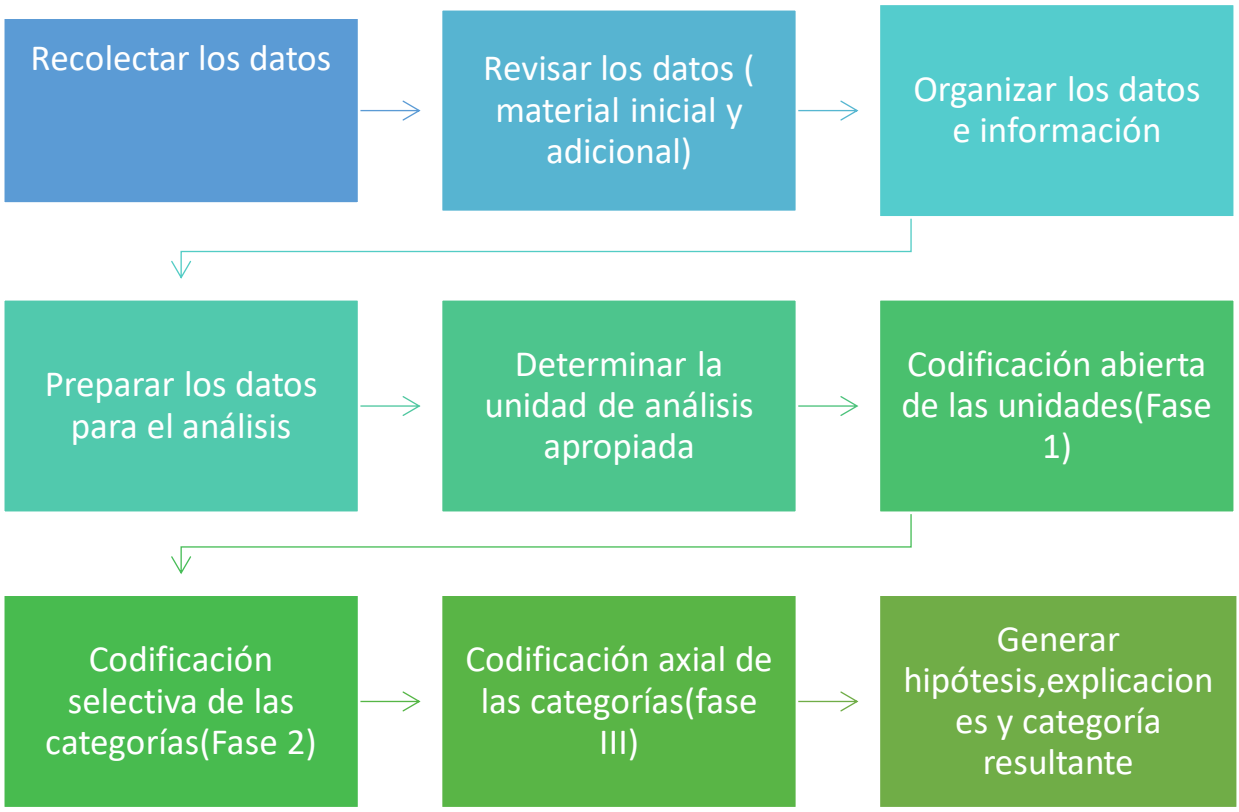
Con los datos recogidos en el punto anterior, se procede con el análisis de los mismos. El análisis de datos consiste en analizar la información recolectada, la cual debe ir ligada con los requerimientos de información identificados con los objetivos definidos en la investigación (Arias, 2012).

El análisis de los datos depende del tipo de datos que se recolecten: cuantitativos (datos numéricos) o cualitativos (narrativas escritas, visuales, auditivas) (Hernández, 2017).

Tal como lo recomienda Hernández (2017) y para efectos de esta investigación, el análisis se basará en la teoría fundamentada a partir de una base de datos, siguiendo el siguiente procedimiento mostrado en la figura 10.

Figura N.º 10

Procedimiento general del análisis para el diseño mixto



Nota: Hernández (2017).

El proceso descrito lleva el procedimiento desde la recolección de los datos hasta las conclusiones que se obtienen al analizar los mismos. Una vez recolectados los datos, se procede a una revisión inicial y organización de la información organizando las fuentes de información en bases de datos confiables para el análisis. Además, se determina la unidad, la cual en la presente investigación será cada respuesta dada por el entrevistado, se identifican nuevas unidades y se categorizan.

Una vez que se tienen las categorías, se trabajará en la identificación de relaciones entre ellas o temas identificados.

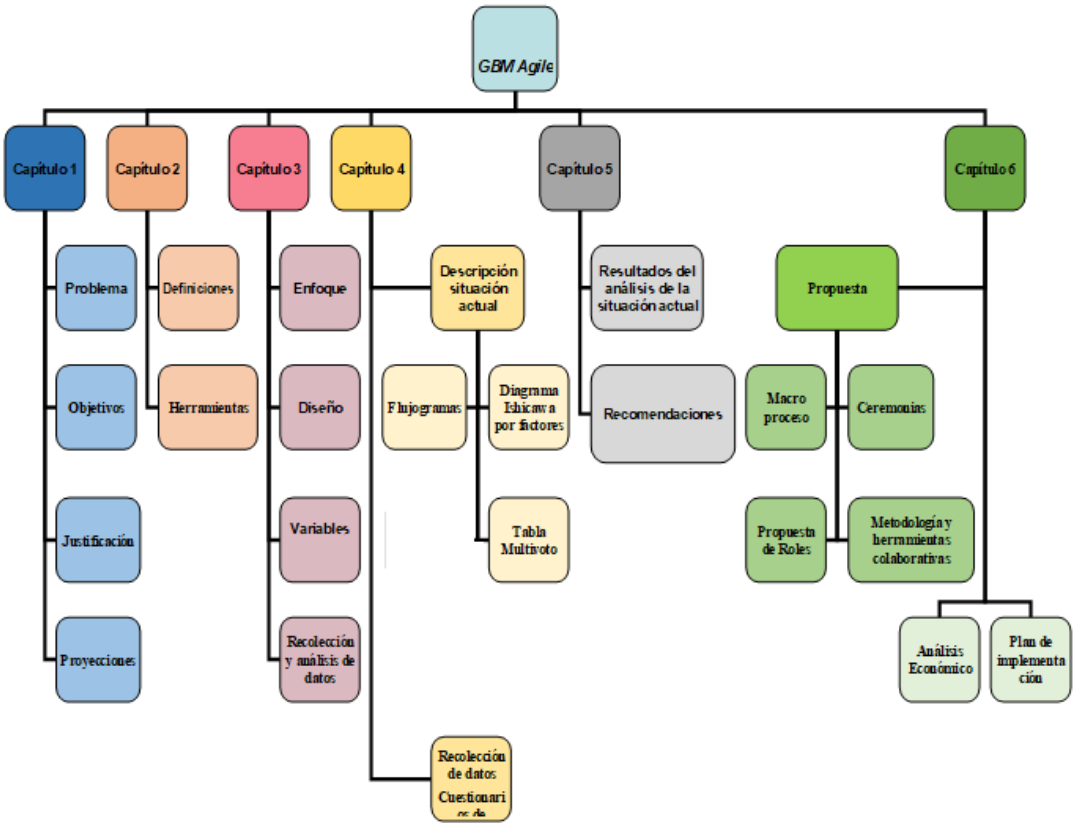
Cronograma

Work Breakdown Structure (WBS)

La estructura de descomposición del trabajo (EDT), también conocida por su nombre en inglés *Work Breakdown Structure* o WBS, es una herramienta fundamental que consiste en la descomposición jerárquica, orientada al entregable, del trabajo que será ejecutado por el equipo de proyecto, para cumplir con los objetivos de este (Heizer & Render, 2009).

Figura N.º 11

WBS (*Work Breakdown Structure*)



Nota: elaboración propia.

El WBS mostrado explica los entregables que se desarrollarán a lo largo de la presente investigación, en el capítulo I se trabaja el problema, los objetivos, la justificación y las

proyecciones; seguidamente, en el capítulo II se trabaja el marco teórico, el cual expone teóricamente los términos fundamentales del presente proyecto.

Una vez completado el marco teórico, se documenta en el capítulo III el marco metodológico, el cual describe la situación actual, población, variables e instrumentos por utilizar, así como la descripción correspondiente a la recolección de datos.

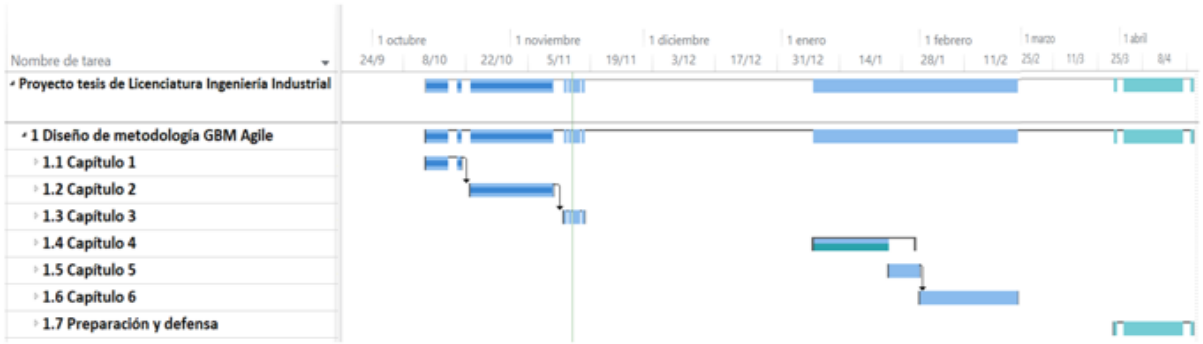
En los capítulos IV y V, se documenta el análisis correspondiente de toda la información recolectada, conclusiones y recomendaciones, quedando para el capítulo 6 la propuesta de diseño de la metodología ágil para GBM.

Diagrama de Gantt

Como herramienta visual para determinar las actividades y cargas de trabajo (Heizer & Render, 2009), se crea el siguiente diagrama de Gantt para la presente investigación.

Figura N.º 12

Diagrama de Gantt



Proyecto tesis de Licenciatura Ingeniería Industrial	136 días	474 horas	vie 13/10/17	mar 24/4/18
1 Diseño de metodología GBM Agile	136 días	474 horas	vie 13/10/17	mar 24/4/18
1.1 Capítulo 1	5,88 días	32 horas	vie 13/10/17	vie 20/10/17
1.2 Capítulo 2	15 días	120 horas	lun 23/10/17	vie 10/11/17
1.3 Capítulo 3	5 días	40 horas	lun 13/11/17	vie 17/11/17
1.4 Capítulo 4	17 días	104 horas	lun 8/1/18	mié 31/1/18
1.5 Capítulo 5	5 días	40 horas	jue 25/1/18	jue 1/2/18
1.6 Capítulo 6	16 días	120 horas	jue 1/2/18	vie 23/2/18
1.7 Preparación y defensa	17 días	18 horas	vie 30/3/18	mar 24/4/18

Nota: elaboración propia.

El diagrama de Gantt muestra el desglose de actividades requeridas para finalizar la investigación, extendiéndose desde octubre 2017 hasta abril 2018, se visualiza la necesidad de 474 horas de trabajo distribuidas en 136 días.

CAPÍTULO IV. ANÁLISIS DE LA SITUACIÓN

La selección de oportunidades de mejoramiento viables para la necesidad de innovación y resolución de problemas existentes en los procesos de una compañía o en este caso de un área de trabajo, se basa en tres aspectos básicos: económicos, técnicos y humanos. Las consideraciones económicas pueden involucrar nuevos servicios, servicios existentes que tienen costos de desarrollo o ventas altos, servicios obsoletos en el mercado o actividades que en el proceso están generando reprocesos y, por consiguiente, aumento en el costo de la prestación del servicio y la insatisfacción de los clientes. Las consideraciones técnicas pueden incluir los procesos, la metodología, prácticas de mejora continua, estándares y mediciones que presenten deficiencias con respecto a la competencia. Los aspectos humanos pueden incluir subutilización de los recursos, falta de capacitación o inadecuado balanceo de cargas.

Con el fin de definir y diseñar la guía metodológica, es necesario realizar un diagnóstico de la situación actual de los procesos y sus prácticas, para esto existen varias herramientas para identificar donde están las oportunidades de mejora que más están afectando al área de Integration GBM. Herramientas como los diagramas de proceso, diagramas de flujo, el diagrama Ishikawa, la matriz EFI y la matriz multivoto son algunas de las herramientas en las que el presente estudio se apoya para la exploración, registro y análisis de la situación actual logrando presentar la información de una manera clara, coherente y concisa.

A continuación, se detalla la información sobre la situación actual del área en estudio recolectada mediante diferentes herramientas de análisis:

Descripción de los procesos actuales

Para conocer la situación actual del área de Integration, se realiza una serie de conversatorios con los recursos que conforman el área, de esta forma, se logra obtener la información necesaria para el entendimiento de la estructura, los procesos y procedimientos actuales para el desarrollo de los requerimientos de *software*. (Apéndice #1. Guía de preguntas para conversatorio).

Una vez realizado este conversatorio, se realiza una revisión de los datos recolectados para realizar un diagnóstico sobre la situación actual, incluyendo un diagnóstico a alto nivel sobre el conocimiento actual en prácticas de desarrollo de *software*, el cual incluye prácticas ágiles de gestión, calidad, análisis y levantamiento de requerimientos y versionamiento.

A continuación, se muestran los resultados obtenidos.

Diagrama de proceso Integration

Actualmente, el área de Integration sigue un proceso tradicional de desarrollo de *software*, la mayoría de las veces denominado Cascada, donde se identifican las etapas mostradas en la figura 13.

Figura N.º 13

Diagrama de proceso general área de Integration

Nota: elaboración propia.

Bajo este esquema, todas las tareas son dependientes unas de otras y no se puede iniciar la segunda etapa si la primera no está concluida. Se visualizan algunos de los roles tradicionales en el desarrollo de *software* como lo son el líder técnico, los desarrolladores, *planners/project manager* y atención al cliente, sin embargo, no siempre ni por todos los recursos se sigue esta metodología tradicional.

Conversatorio del equipo Integration

El área de Integration actualmente se encuentra conformada por por equipos denominados Integration Core (IC), Business Activity (BA) y Software solutions (SO), 40 recursos en total con tres líderes técnicos, quienes son los responsables de atender los diferentes proyectos y solicitudes de los clientes.

Se cuenta con el apoyo de un rol denominado *planner* o *project manager*, quienes son los responsables de velar por la asignación de los recursos, a las diferentes solicitudes que llegan por

medio de este rol. A continuación, se expone el desglose por equipo de la información recolectada.

Equipo Integration Core (IC)

Tabla N.º 3

Hallazgos Equipo IC

Estructura administrativa	Actualmente, la tarea de gestionar al equipo se encuentra bajo una distribución del 60/40, el 40% lo administra el líder técnico y el restante lo gestiona el *planner/project manager*.
Roles actuales	El equipo IC está conformado por 10 recursos y un líder técnico. Distribuidos bajo los siguientes roles: *Release management* (1), rol recientemente incluido al proceso. Desarrolladores/ Calidad (9), Líder técnico (1),
Inventario de las solitudes realizadas	Se cuenta con catálogo de servicios en formato Excel, su nivel es muy general y comúnmente lleva el mismo nombre de los casos de uso.
Productos que desarrollan	Integración del BPM Bpels WAS Cas Iron MQ

Gestión de recursos	Actualmente, los recursos son asignados según sus destrezas y habilidades, los recursos que se encuentran dentro del grupo de *Recursos Flotantes*, el trabajo se lo asignan las *planners o project managers*, esto para la atención de tiquetes. Para la asignación de los lotes y requerimientos en sí, el líder técnico asigna el trabajo de acuerdo con la carga de trabajo y las destrezas de los recursos. Los requerimientos o tiquetes del producto WAS presentan una excepción, ya que son auto planificados por el recurso que los desarrolla y se desconoce la producción y asignación real del trabajo.
Tool de gestión y administrativas	El equipo IC carece de una herramienta de gestión, actualmente utilizan un registro de tareas en formato Excel. Adicional a este registro, se cuenta con un calendario de vacaciones, fechas importantes y ausencias.
Métricas	Actualmente, el equipo IC no cuenta con métricas que midan los procesos y sus rendimientos de forma oficial.
Plantillas	La única plantilla o artefacto utilizada hoy en día es una bitácora semanal histórica de los servicios, mencionada como *tool* de gestión.
Seguimiento	Actualmente, no se realiza ningún tipo de sesión de seguimiento.
Tipo de entregables	El formato de los entregables que realiza el equipo IC consiste en un documento de servicio con formato GBM, este documento es una especie de manual para el uso del servicio.
Desperdicios (Retrabajo y tiempo muertos)	No se tiene información.

Cantidad de defectos	No se tiene información
Producción mensual	No se tienen registros

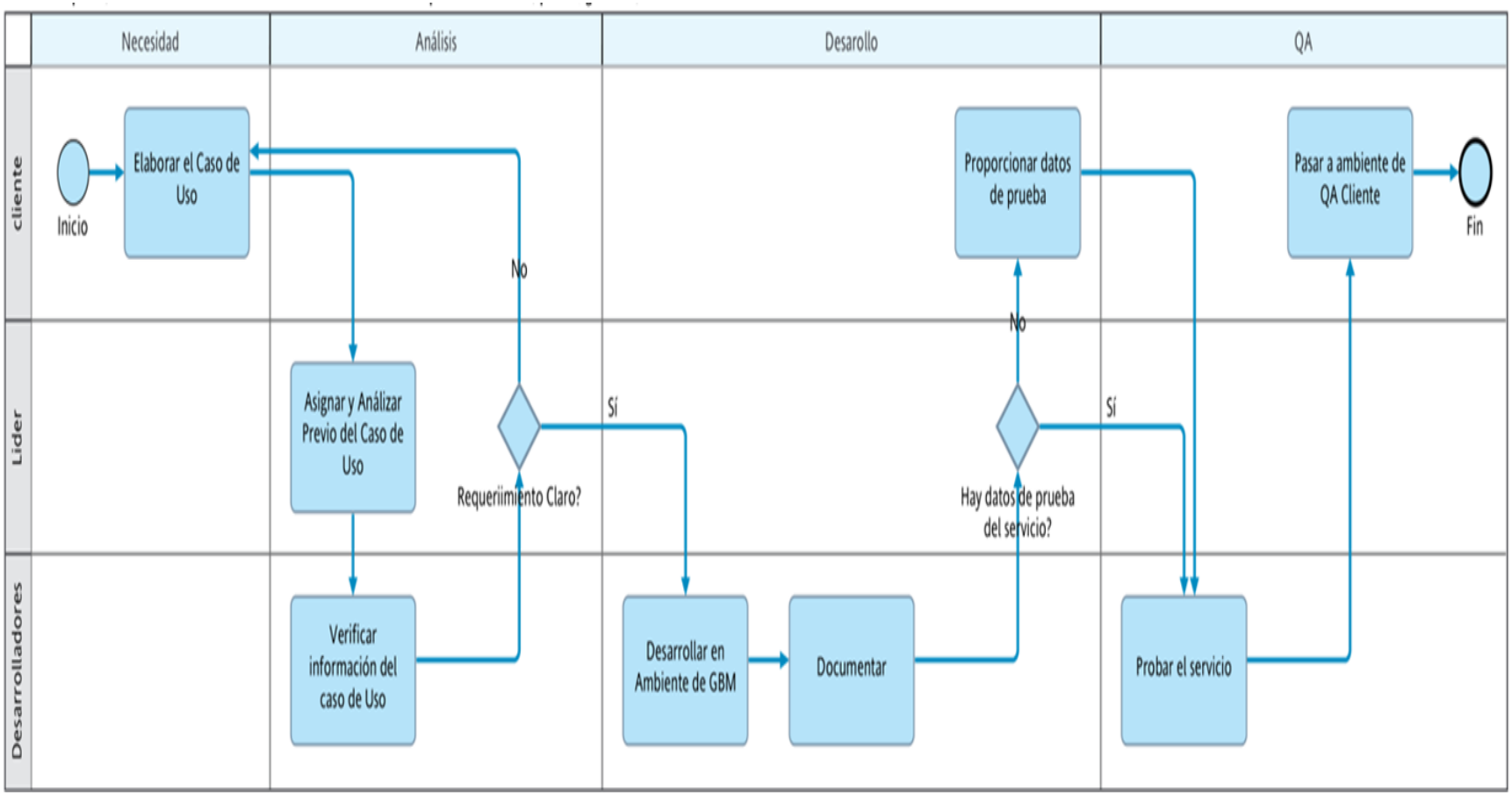
Nota: elaboración propia.

Diagrama de flujo del equipo Integration Core.

Se presenta el diagrama de flujo actual del equipo IC, el cual muestra la participación de tres roles y cuatro etapas.

Figura N.º 14

Diagrama de flujo Equipo Integration Core



Nota: elaboración propia.

Características actuales del proceso.

El proceso inicia cuando el cliente le envía un caso de uso al líder técnico, generalmente, este envío es realizado vía correo electrónico, actualmente no se cuenta con una plantilla estándar para el levantamiento de requerimientos. Cada requerimiento es considerado un caso de uso, no está estandarizado el uso de un identificador en ninguna de las partes (GBM-Cliente).

El análisis del requerimiento por parte del líder técnico antes de su desarrollo no se da en todos los casos, si durante el análisis se presentan dudas, las mismas son aclaradas con el cliente. Una vez claro el requerimiento, el equipo de desarrollo ejecuta sus tareas en el ambiente de desarrollo de GBM y una vez finalizados los desarrollos, los mismos son trasladados a un ambiente de calidad donde realizan las pruebas internas (en algunas ocasiones no se cuenta con los datos de pruebas, por lo que la realización de las mismas se ve imposibilitada parcial o totalmente).

Una vez realizadas las pruebas internas, los desarrollos son trasladados al ambiente de calidad del cliente, quien terceriza las pruebas con otro proveedor. La decisión de realizar el pase a producción es propia del cliente, en ocasiones los desarrollos se mantienen en el ambiente de calidad del cliente y no pasan a producción por largos periodos de tiempo.

Se evidencia un desbalanceo de las cargas de trabajo durante el proceso, existen recursos muy cargados, mientras otros están en tareas de investigación y desarrollo, sin haber sido completamente capacitados.

Descripción del proceso de recepción de solicitudes.

Actualmente los requerimientos se categorizan de la siguiente manera:

a-Soporte: aquellos requerimientos del día a día que conforman el mantenimiento simple de las aplicaciones.

b- Lotes de producción: aquellos servicios nuevos y mantenimiento de servicios.

En algunas ocasiones, el requerimiento no es levantado de forma correcta, la mayoría de las solicitudes llegan en lotes y en algunos casos, se mezclan los servicios nuevos con requerimientos de cambios.

Cuando se presentan requerimientos de cambios, los cambios nuevos se adjuntan a la versión inicial del documento y se pone la versión del cambio del documento actual, sin embargo, no se encuentra estandarizado este proceso, por lo que no siempre se aplica.

Proceso de planificación de la producción.

La planificación de la producción la realizan mediante el cálculo de holguras, estimando un poco más de tiempo para las tareas, esto debido a que la prioridad es soporte en lugar de los lotes. Se planifica por requerimiento, dado que no se tiene mayor visibilidad. Cuando llegan varios requerimientos, el líder técnico construye un *backlog* (Listado) en el Excel de registro de tareas.

Equipo Business Automation

Tabla N.º 4

Hallazgos Equipo BA

Estructura administrativa	El equipo BA actualmente está conformado por 12 recursos incluido el líder técnico.
Roles actuales	Actualmente, toda la gestión de los recursos recae sobre el líder técnico, es el único rol mapeado formalmente.
Inventario de solicitudes	Actualmente, no se cuenta con un inventario de las solicitudes recibidas.
Productos que desarrollan	BPM Integration Bus MQ Broker Sterling Conect Direct Transformation Xtender BPELs IBM Data Power

	Cas Iron IBM file Gateway Data Power Operation dashboard Sterling BtB Open Legasi
Gestión de recursos	No se tiene la categorización de los expertos, actualmente se está montando una matriz con esta información. La atención de las solicitudes o tiquetes las atienden los recursos "flotantes" y la asignación es por demanda.
Herramientas de gestión y administración	El líder técnico no utiliza ninguna herramienta de gestión para el seguimiento de los desarrollos.
Métricas	El equipo recién incluyo métricas de la práctica de *release management*, son las únicas con las que se cuenta actualmente.
Plantillas	Se utiliza solamente un reporte de horas para un proyecto puntual, no es una práctica estandarizada para todos.
Seguimiento	No se realizan sesiones de seguimiento, hubo en su momento la iniciativa de realizar las sesiones diarias que proponen las prácticas ágiles, sin embargo, el equipo no logra mantener su realización.
Tipo de entregables	El equipo maneja como entregable un documento de pase a producción y un documento técnico que no es estandarizado para todos los proyectos.
Desperdicios (Retrabajo y tiempo muertos)	No se tiene información

Cantidad de defectos	No se tiene información
Producción mensual	Al no tener registros suficientes, actualmente no se lleva control de la capacidad de producción del equipo.

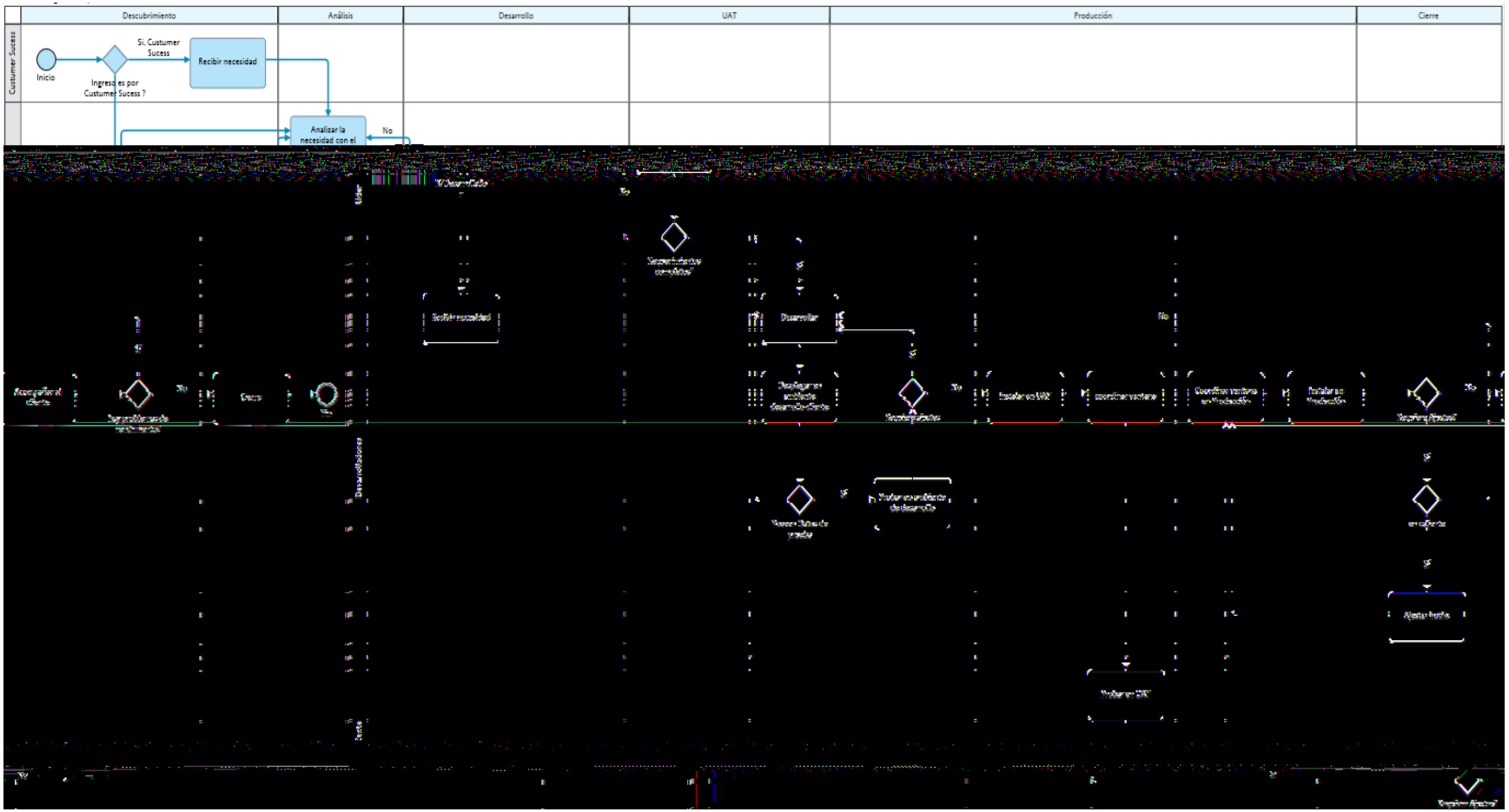
Nota: elaboración propia.

Diagrama de flujo del equipo Business Automation (BA).

Se presenta, a continuación, el diagrama de flujo del equipo BA, con cuatro roles mapeados y seis etapas desde el inicio del proceso hasta su final.

Figura N.º 15

Diagrama de flujo Equipo Business Automation



Nota: elaboración propia.

Características del proceso actual.

El proceso inicia cuando el cliente envía un requerimiento, el mismo es estimado a muy alto nivel y documentado en una plantilla denominada Boom, más adelante en el proceso el requerimiento es analizado con mayor detalle, esto es parte de un proceso de preventa. Una vez que el requerimiento es recibido en GBM, ya confirmada su venta, el mismo pasa por un recurso bajo el rol denominado Customer Sucess, quien se encuentra en contacto con el cliente directo. Esta persona notifica sobre la nueva solicitud al líder técnico por correo o por llamada (la información que suministra viene muy general).

El líder técnico es notificado de forma tal que pueda reservar un espacio para clarificar la necesidad. Si durante la sesión hay recursos técnicos, las dudas son aclaradas, de lo contrario, las dudas son enviadas por correo electrónico para conseguir la información internamente. El equipo no habla directamente con los proveedores del cliente, la mayoría del tiempo es el cliente el intermediario entre sus otros proveedores y el equipo de GBM.

Los desarrollos inician en el ambiente del cliente, el código se mantiene en la máquina del desarrollador y para probar se usan los datos de prueba que entrega el cliente. Generalmente solo se prueba el *happy path* (camino feliz). Una vez realizadas las pruebas, se trabaja en la documentación y se entrega lo desarrollado para pruebas UAT (User Aceptance Test) con IBM.

El área de calidad del cliente ejecuta las pruebas respectivas, si existen defectos, se devuelve el requerimiento al proceso de construcción, esta devolución se realiza por medio de un correo o un chat. Solo en casos esporádicos se pide documentación. Una vez que el requerimiento pasa el proceso de pruebas, se genera el documento de pase a producción, se coordina con IBM/GBM y se brinda acompañamiento post-puesta en producción.

Equipo Software Solutions (SO)

Tabla N.º 5

Hallazgos Equipo SO

Estructura administrativa	82/18, el 18% se encuentra asignado a cliente, el 82% del recurso se considera recurso flotante.
Roles actuales	Líder técnico Representante del cliente Planners / Project Manager Desarrolladores / Calidad
Inventario de solicitudes	Actualmente no se cuenta con un inventario de las solicitudes recibidas.
Productos que desarrollan	Broker Wsrr Cas Iron Data power Sterling MQ.
Gestión de recursos	Los recursos son asignados por destrezas y disponibilidad. Todos los recursos son gestionados por los *planners/project manager*, ya sea que la solicitud ingrese por Service Desk o por el representante del cliente, las *planners* son las que realizan la asignación respectiva.
Tools de gestión y administración	Carencia de herramienta de gestión, líderes solamente recomiendan cuál recurso podría atender según sus destrezas, por lo que la mayor

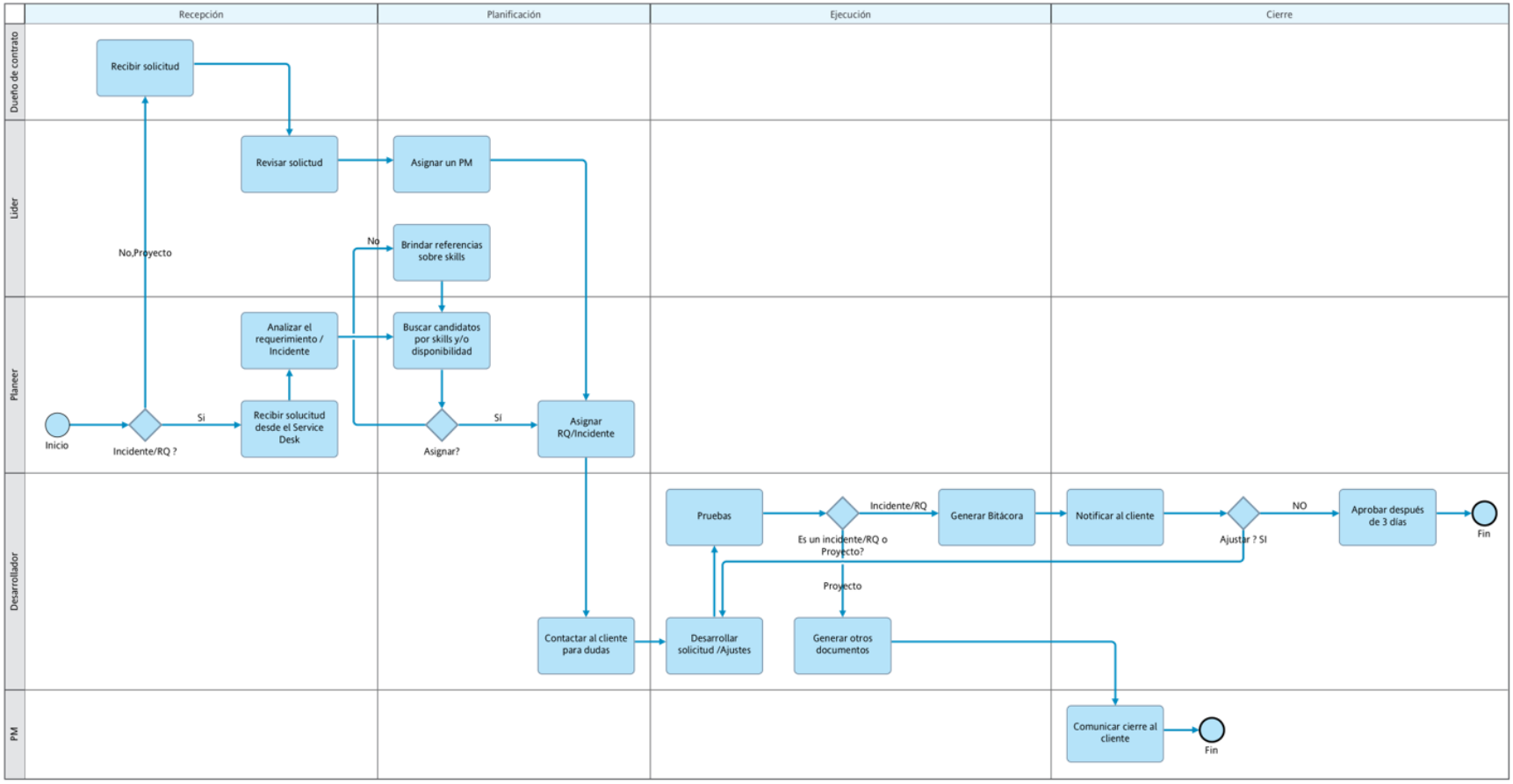
	gestión la tienen los *planners/project manager*.
Métricas	Actualmente no hay métricas que midan los procesos.
Plantillas	Cuando es un proyecto, hay una plantilla para el análisis y diseño. Existe una plantilla para estimar tiempos de desarrollo. Bitácora para RQ e incidentes.
Seguimiento	Dependiendo del cliente se realizan ceremonias ágiles como el *daily*.
Tipo de entregables	Bitácora, plantilla de análisis y diseño, dependiendo del cliente deben utilizar las plantillas que ellos soliciten.
Desperdicios (Retrabajo y tiempo muertos)	La demanda genera muchos tiempos muertos como las esperas por ambientes, esperas de datos de prueba, permisos, accesos, servicios. En la bitácora se contabilizan todos los tiempos, los tiempos de espera van sumados en el reporte y se suman los pendientes del cliente. Si se dan atrasos por parte de GBM, se solicita apoyo a lo interno de GBM para resolver.
Cantidad de defectos	No se tiene información
Producción mensual	No se tiene información

Diagrama de Flujo Equipo SO.

Se presenta el diagrama de flujo del equipo SO con cinco roles identificados y cuatro etapas del proceso.

Figura N.º 16

Diagrama de flujo Equipo SO



Nota: elaboración propia.

Proceso actual.

El proceso inicia cuando las *planners* reciben las solicitudes, luego las analizan y buscan los recursos con las habilidades necesarias para atender la solicitud; para poder determinar el recurso ideal, se apoyan en el líder técnico, una vez identificado el recurso verifican su disponibilidad y le asignan la solicitud.

El desarrollador generalmente debe realizar un levantamiento del requerimiento, para luego realizar el análisis y desarrollo correspondiente. Una vez que el requerimiento es desarrollado, los mismos ingenieros desarrollan, prueban y realizan la entrega al cliente.

Se notifica al cliente, quien cuenta con tres días para la revisión correspondiente, si el cliente lo requiere se ajusta y se vuelve a realizar el ciclo de pruebas, si no se recibe respuesta del cliente durante los tres días, el desarrollo se da por aprobado (Aplica para RQS o incidentes). Al finalizar el desarrollo de requerimientos o incidentes, se genera una bitácora de lo trabajado, en la ejecución de un proyecto es necesaria la generación de otros documentos.

Descripción de proceso de recepción de solicitudes.

Las solicitudes puedes ingresar desde dos vías:

Service Desk: utilizado para requerimientos pequeños, cliente va directo al Service Desk, las *planners* son las encargadas de gestionar lo que ingresa por esta vía. Los insumos que vienen desde el Service Desk usualmente son correos, con escasa información.

Representante del cliente: cliente hace la solicitud a su representante, se revisa la solicitud con el líder técnico y se evalúa su complejidad y tamaño.

Si es un proyecto se asigna un *project manager* y este es el encargado de comunicar a las *planners* la solicitud e inicia el proceso. Si es algo pequeño, el representante del cliente baja el requerimiento a las *planners/project managers* y estas realizan la asignación .

Proceso de planificación de la producción.

La producción es planificada por las *planners,* de acuerdo con la disponibilidad.

Diagnóstico madurez actual en prácticas de desarrollo de *software*

Con el fin de conocer la madurez y el conocimiento actual de los equipos en prácticas de desarrollo de *software*, se realiza el cálculo de una muestra representativa de 26 recursos

(Apéndice #2. Cálculo de muestra) pertenecientes al área en estudio, sin embargo, por solicitud de la empresa, se realiza el cuestionario al total de recursos del área de Integration (40 personas). Esto ya que se contaba con la disponibilidad de tiempo de los recursos y del analista.

Figura N.º 17

Cálculo de muestras para poblaciones finitas

Cálculo de Muestras para Poblaciones Finitas

INGRESO DE PARAMETROS	
Tamaño de la Población (N)	40
Error Muestral (E)	0,05
Proporción de Éxito (P)	0,05
Proporción de Fracaso (Q)	0,95
Valor para Confianza (Z)	1,96

	Tamaño de Muestra
Muestra Optima	26

Nota: elaboración propia.

Con cada recurso se llevó a cabo sesiones donde se revisaban diversos cuestionarios sobre las prácticas de levantamiento de requerimientos, Aseguramiento y Control de Calidad, *release management* y Gestión por medio de la cuál al promediar las respuestas se evidencia el estado del conocimiento y/o madurez de cada equipo en las prácticas anteriormente mencionadas. (Apéndice # 3 Cuestionarios Prácticas desarrollo de *software*)

Para cada práctica se definen los siguientes criterios de evaluación:

Levantamiento de requerimientos.

Se evalúa, mediante una serie de preguntas, el nivel de tres criterios principales de la práctica:

1- Nivel de madurez del equipo en aspectos fundamentales del levantamiento de requerimientos: para conocer el porcentaje de la madurez en esta práctica, se consulta a los entrevistados sobre las etapas que comprenden el proceso actual, el tipo de análisis que se realiza una vez que se presenta una necesidad, la existencia de la definición de un mínimo producto viable, los roles actuales, plantillas utilizadas y la forma en que actualmente se manejan las dependencias.

2- Nivel de claridad de los requerimientos proporcionados a los desarrolladores: por medio del cuestionario, se logra conocer si los requisitos son analizados por el líder técnico, con el objetivo de velar por los posibles riesgos o impedimentos relacionados con los requerimientos, conocer si los desarrolladores cuentan con toda la información necesaria para desarrollar y si el equipo entiende todos los requerimientos que son proporcionados.

3- Nivel de conocimiento y práctica actual de las metodologías ágiles en el levantamiento de requisitos: para determinar el nivel de cada equipo, se indaga sobre el conocimiento de la definición de historias de usuario, uso de las historias de usuario como formato de redacción de requerimientos, la existencia de criterios de aceptación y el manejo de identificadores (ID) por requerimiento.

Release management.

Para conocer la madurez actual en la práctica de *release management* (versionamiento), el cuestionario se centra en identificar los niveles de madurez de los siguientes criterios:

1- Nivel de conocimiento sobre prácticas de *release managemen:* para determinar el nivel de conocimiento, se evalúa si el equipo cuenta actualmente con conocimientos básicos de la gestión de versiones, si se conocen los elementos fundamentales de un repositorio, si se maneja alguna experiencia en el uso de la herramienta GIT y si se cuenta con proyectos en repositorios en este momento.

2- Nivel de estandarización de metodologías establecidas sobre *release management* en el equipo: este nivel se obtiene mediante la consulta al equipo sobre si actualmente cuentan con una herramienta para realizar los *release* de sus proyectos, si se manejan estándares de nomenclatura para sus objetos, si existe un estándar para la creación de repositorios, si cuentan con un control de notificaciones hacia el cliente cada vez que realizan un *release* y si visualizan algún inconveniente para utilizar metodologías de *release.*

3- Nivel de planificación actual para el versionamiento: el nivel de planificación se evalúa básicamente por medio de dos consultas simples, las cuales intentan conocer si se tiene una hora específica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio y si se planifican los *releases* desde el inicio del proyecto.

Aseguramiento y control de la calidad.

Para determinar la madurez actual del equipo en la práctica de aseguramiento y control de calidad, se realiza una evaluación basándose en los siguientes criterios:

1- Nivel de involucramiento temprano de calidad en el proceso de desarrollo de *software*: para determinar este nivel, las consultas se basaron en conocer si actualmente existe un involucramiento temprano de los ingenieros de Calidad en sus proyectos, si los requerimientos son analizados por los ingenieros de Calidad, con el objetivo de velar por los posibles riesgos, impedimentos, necesidades especiales o identificación de datos para pruebas; si los ingenieros de Calidad cuentan con toda la información necesaria para desarrollar sus pruebas y si se identifica en etapas tempranas la necesidad de pruebas no funcionales (rendimiento, recuperación, etc.).

2- Nivel de madurez del departamento en las prácticas de calidad: para la definición de este nivel, el cuestionario se basa en variables como la estandarización de procesos, las prácticas de calidad informales, la gestión del administrador de proyectos, los procesos de cierre y la claridad en los criterios de aceptación, revisando si actualmente el equipo cuenta con un proceso definido y estándar de desarrollo, si existe un proceso definido y estándar de calidad y de pruebas en el equipo, si se diseñan, ejecutan y documentan pruebas unitarias; si se cuenta con criterios de aceptación para cada requerimiento, si los criterios de aceptación son 100 % claros, sin ambigüedades y alineados para asegurar el cumplimiento de la necesidad del negocio; si se documentan los cambios realizados por el cliente una vez iniciado el proyecto, si se especifica el impacto de los cambios solicitados por el cliente y si se diseña un plan de pruebas y casos de pruebas para cada proyecto.

3- Nivel de la madurez del cierre del proceso de calidad y el uso de indicadores: este nivel se define evaluando variables como las prácticas informales de calidad, procesos de cierre, rol del ingeniero de Calidad y uso de indicadores de calidad, mediante la averiguación sobre si actualmente existe un estándar de nomenclatura para los objetos creados, si se documentan e informan los defectos utilizando una herramienta de seguimiento de defectos o reportes en documentos de Office, si se monitorea la resolución de defectos y los esfuerzos necesarios para ello, si se realizan pruebas no funcionales en sus desarrollos de forma proactiva y reactiva, si se documentan e informan los resultados de las pruebas no funcionales realizadas, si se documenta un informe final de calidad con un formato determinado, si se utilizan kpi's en el equipo, con el

fin de contar con información para mejora continua y si existe un rol de calidad claramente definido.

A continuación, se muestran los resultados obtenidos por equipo según práctica y criterios (Apéndice # 3 Resultados de evaluación de variables por criterios).

Equipo Integration Core (IC).

Tal como se presentó en la sección anterior del presente documento, el equipo IC se dedica al desarrollo de integraciones BPM y BPEl, se encuentra conformado por 11 recursos, a quienes se les realiza el cuestionario por práctica establecidos como instrumentos del presente estudio.

Práctica de levantamiento de requerimientos.

En general, el equipo IC muestra una madurez de un 32 % en la práctica de levantamiento de requerimientos, desglosado de la siguiente manera en la tabla 6.

Tabla N.º 6

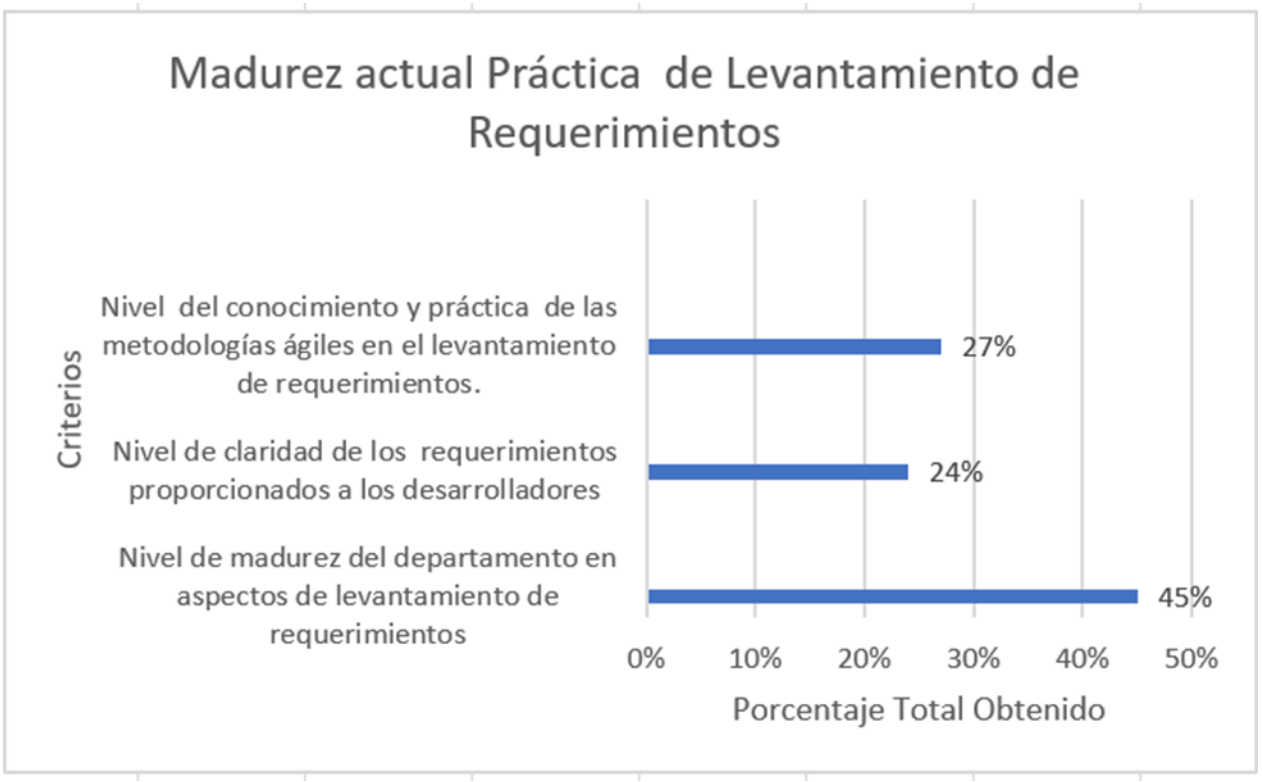
Madurez actual práctica de levantamiento de requerimientos Equipo IC

Madurez actual Práctica de Levantamiento de Requerimientos Equipo IC	
Nivel de madurez del departamento en aspectos de levantamiento de requerimientos	45%
Nivel de claridad de los requerimientos proporcionados a los desarrolladores	24%
Nivel del conocimiento y práctica de las metodologías ágiles en el levantamiento de requerimientos.	27%

Nota: elaboración propia.

Figura N.º 18

Madurez actual práctica de levantamiento de requerimientos Equipo IC



Nota: elaboración propia.

Como se puede observar, el equipo IC cuenta con un nivel de madurez en aspectos de levantamiento de requerimientos de un 45 %, esto como resultado de las opiniones de los 11 recursos que conforman el equipo, donde un 100 % considera que sí existe la etapa definida para el levantamiento de requerimientos, un 82 % considera que sí se analiza el problema antes de dar solución, un 64 % está de acuerdo con que los mismos recursos que levantan el requerimiento son quienes lo desarrollan, solo un 27 % de los recursos considera que se tiene un formato estandarizado para la documentación de los requerimientos y el 100 % está de acuerdo con que actualmente no se define un mínimo producto viable ni se cuenta con un método para el manejo de las dependencias entre los requerimientos.

El nivel de claridad de los requerimientos para el equipo IC se encuentra en un 24 %, para este criterio solamente un 45 % registra que los requisitos son analizados por el líder técnico del equipo, solo un 18 % acepta tener toda la información requerida para el desarrollo del

requerimiento y en una mínima representación, solamente el 9 % dice tener claro lo que deben realizar en cada requerimiento.

Con respecto a las prácticas ágiles en el levantamiento de requerimientos, el equipo muestra un nivel de conocimiento de un 27 %, donde un 45 % de los recursos dice conocer lo que es una historia de usuario, aun así el 100 % considera que no se está utilizando la historia de usuario para la documentación de los requerimientos; un 36 % mantiene que los requerimientos actuales sí contienen criterios de aceptación, mientras que solamente un 27 % reconoce un identificador en el documento de requerimiento que se usa actualmente.

Práctica Release management.

Con respecto a las prácticas de *Release management*, el equipo IC muestra un nivel promedio de un 53 %, a continuación, se puede observar el desglose respectivo en la tabla 7.

Tabla N.º 7

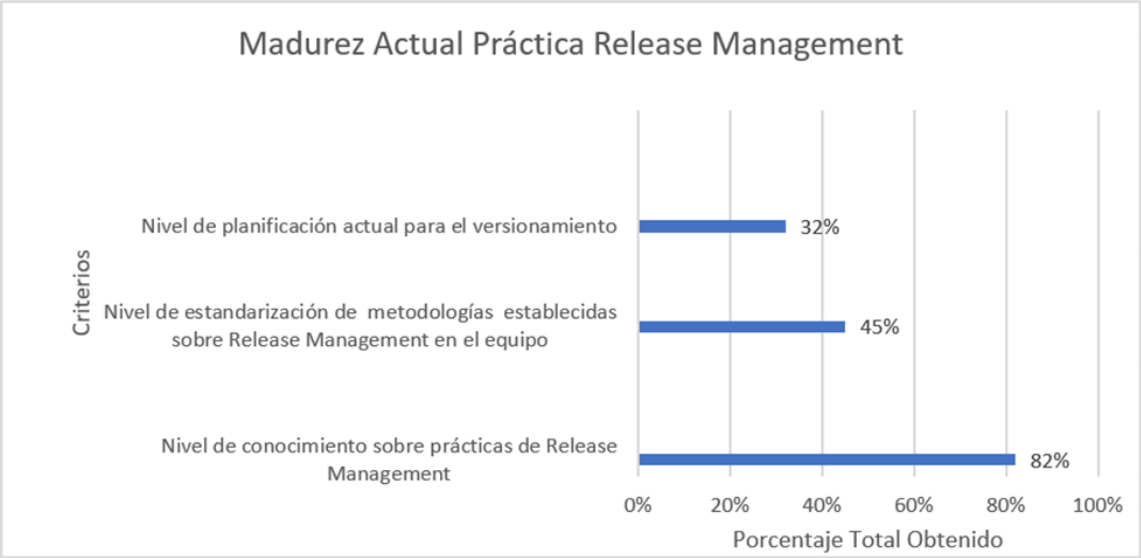
Madurez actual práctica de *Release management* Equipo IC

Madurez actual Práctica Release Management Equipo IC	
Nivel de conocimiento sobre prácticas de Release Management	82%
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	45%
Nivel de planificación actual para el versionamiento	32%

Nota: elaboración propia.

Figura N.º 19

Madurez actual práctica de *Release management* Equipo IC



Nota: elaboración propia.

Al aplicar el cuestionario a los 11 recursos que integran el equipo IC, se encuentra que el conocimiento actual sobre prácticas de *release* está en un 67 %, donde el 82% considera tener conocimientos básicos del proceso de gestión de versiones y elementos fundamentales de un repositorio y un 36 % tiene conocimiento del *software* más recomendado para esta práctica en los primeros pasos de su implementación denominado GIT.

Actualmente, el equipo IC muestra una estandarización metodológica de un 45 %, bajo este criterio, un 64 % ya maneja proyectos en repositorios y un 45 % lo realiza mediante una herramienta automatizada. Sobre los estándares, se encuentra que un 55 % utiliza estándares para la nomenclatura de sus objetos, mientras un 36 % dice manejar un estándar para la creación de repositorios. Un 45 % considera que existe actualmente un control de notificaciones hacia el cliente cada vez que realizan un *release* y un 27 % cree que existen inconvenientes en este momento para establecer una metodología de *release*, donde mencionan entre estos la falta de herramienta, formatos y roles claramente establecidos.

El nivel de planificación de los *releases* se encuentra en un 32 %, donde la mayoría (36%) considera que sí maneja una hora específica para subir el desarrollo de cambios o de nuevas

funcionalidades al repositorio y solamente un 27 % considera haber realizado una planificación de los *releases* al iniciar un proyecto.

Práctica de aseguramiento y control de calidad.

En prácticas de aseguramiento y control de la calidad, según la evaluación realizada, el equipo IC presenta un nivel de madurez de un 40 %, comprendido de la evaluación de los tres criterios. El desglose se muestra en la tabla 8.

Tabla N.º 8

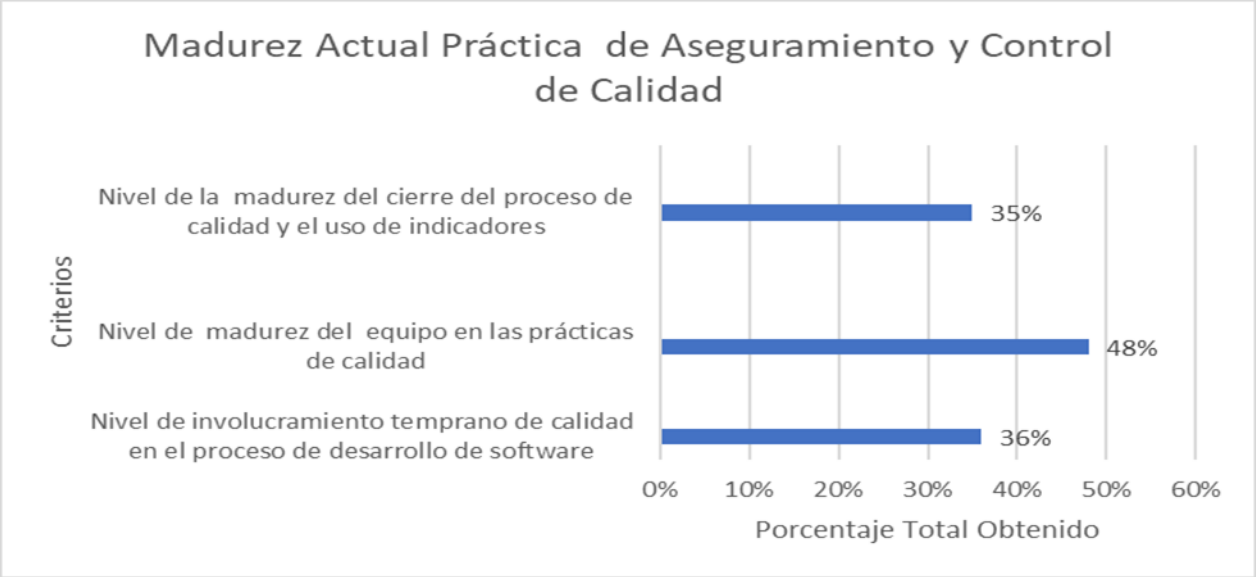
Madurez actual práctica de aseguramiento y control de calidad Equipo IC

Madurez actual Práctica Aseguramiento y Control de Calidad Equipo IC	
Nivel de involucramiento temprano de calidad en el proceso de desarrollo de software	36%
Nivel de madurez del equipo en las prácticas de calidad	48%
Nivel de la madurez del cierre del proceso de calidad y el uso de indicadores	35%

Nota: elaboración propia.

Figura N.º 20

Madurez actual práctica de Aseguramiento y Control de calidad



Nota: elaboración propia.

Según los resultados obtenidos, la madurez del equipo en las prácticas de calidad se encuentra en un 48 %, donde:

91 % manifiesta que existe un proceso definido y estándar de desarrollo en su unidad.

73 % considera que existe un proceso definido y estándar de calidad en su unidad.

Sin embargo, todos los recursos (100%) consideran que no se ha podido establecer el proceso definido de pruebas en su equipo.

El 36 % de los recursos afirma que actualmente sí se diseñan, ejecutan y documentan pruebas unitarias y que los requerimientos sí cuentan con criterios de aceptación. Un 82 % de los recursos considera que cuando los hay, los criterios de aceptación son 100 % claros, sin ambigüedades y alineados a asegurar el cumplimiento de la necesidad del negocio.

Un 45 % afirma que actualmente se documentan los cambios realizados por el cliente, una vez iniciado el proyecto y se diseña un plan de pruebas para cada proyecto. Un 73 % considera que sí se especifica el impacto de los cambios solicitados por el cliente. Finalmente, un 27 % considera que sí se diseñan casos de pruebas en cada proyecto y todos los recursos afirman no manejar un estándar de nomenclatura para los objetos creados.

El nivel de involucramiento del área de calidad desde etapas tempranas del proyecto se encuentra en un 36% , bajo este criterio, se encuentra que un 64 % sí considera que se da un involucramiento temprano de los ingenieros de calidad en sus proyectos, un 36 % confirma el análisis de los requerimientos por parte del ingeniero de Calidad , un 18 % considera que los ingenieros de Calidad cuentan con toda la información necesaria para desarrollar sus pruebas y un 27% considera que cuando se da el involucramiento temprano, se logra identificar la necesidad de pruebas no funcionales (rendimiento, recuperación, etc.).

Se determina también que el nivel de madurez en el proceso de cierre y el uso de indicadores se encuentra en un 35 %, este bajo nivel lo respalda el equipo al manifestar que solamente el 36 % documenta e informa los defectos utilizando una herramienta de seguimiento de defectos o reportes en documentos de office.

Un 45 % afirma que se monitorea la resolución de defectos y los esfuerzos necesarios para ello, un 27 % realiza pruebas no funcionales en sus desarrollos de forma proactiva, mientras que un 64 % las realiza de forma reactiva. El equipo manifiesta que, en el caso de que se apliquen pruebas no funcionales, solo un 64 % de los resultados se documentan e informan. Solamente un 45 % de los recursos afirman contar con un informe final de calidad y todos los recursos confirman no contar con Kpis o indicadores que ayuden a la mejora continua.

Según los resultados obtenidos en las prácticas evaluadas con el Equipo IC, se evidencia una falta de estandarización y de definición clara de los procesos y formatos, así como una falta de claridad en los roles y sus responsabilidades, no existe una metodología clara a seguir, ninguna de las prácticas logra un nivel mayor al 55 %, lo que probablemente implica una falta de definición, capacitación y *coaching*.

La calidad de los entregables se puede estar viendo comprometida, debido a la falta de claridad en los roles, falta de involucramiento temprano de un área de calidad consolidada con el conocimiento necesario para asegurar el cumplimiento de la práctica. Si bien es cierto, los recursos tienen idea de las buenas prácticas, sin embargo, el equipo no ha encontrado la manera de estandarizarlas y lograr su implementación, se evidencia falta de liderazgo y falta de comunicación entre los dos recursos responsables de la gestión.

Equipo Business Automation (BA).

El equipo BA actualmente está conformado por 12 recursos y se centran en el desarrollo de una amplia gama de productos de gran importancia para el área en estudio.

Práctica de levantamiento de requerimientos.

Se logra realizar el cuestionario con todos sus recursos, dando como resultado una madurez de un 33 %.

Tabla N.º 9

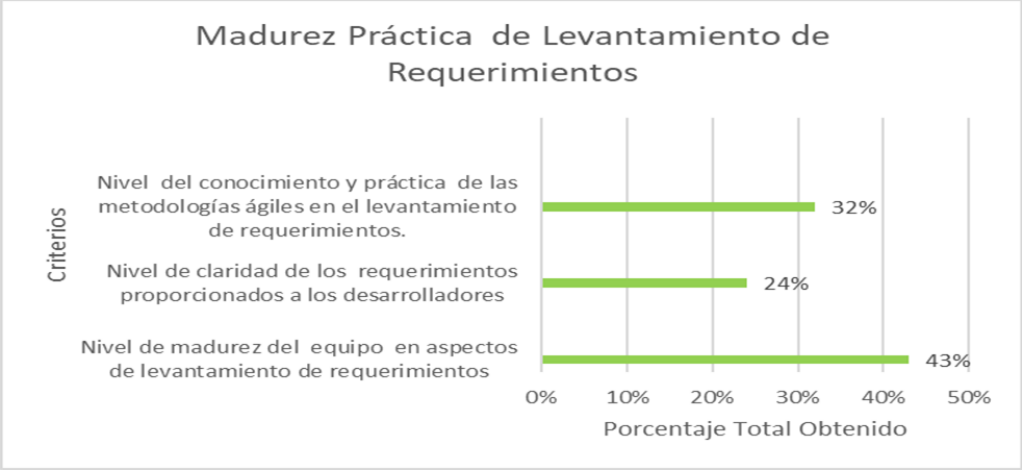
Madurez actual práctica de levantamiento de requerimientos Equipo BA

Madurez actual Práctica Levantamiento de Requerimientos Equipo BA	
Nivel de madurez del equipo en aspectos de levantamiento de requerimientos	43%
Nivel de claridad de los requerimientos proporcionados a los desarrolladores	24%
Nivel del conocimiento y práctica de las metodologías ágiles en el levantamiento de requerimientos.	32%

Nota: elaboración propia.

Figura N.º 21

Madurez actual práctica de levantamiento de requerimientos Equipo BA



Nota: elaboración propia.

Actualmente y según la evaluación realizada, el equipo BA cuenta con una madurez en aspectos de levantamiento de requerimientos de un 43 %, dentro de este criterio se identifica que el 82 % de los recursos considera que sí existe una etapa definida para el levantamiento de los requerimientos. Se encuentra en la evaluación una misma opinión de todo el equipo (100 %), donde se afirma que actualmente no se está identificando un mínimo producto viable y donde se manifiesta que los recursos que están levantando los requerimientos son los mismos que los desarrollan, un 53 % afirma contar con un mismo formato para el levantamiento de requerimientos en todos los proyectos, un 65 % afirma contar con un método para el manejo de las dependencias y un 59 % afirma que se realiza un análisis del problema previo a su solución.

El nivel de claridad de los requerimientos para el equipo BA se determina en un 24%, el 100 % de los recursos considera que el líder técnico no está revisando los requerimientos, solo el 41 % de los recursos considera que los desarrolladores cuentan con toda la información para desarrollar los requerimientos y un 29 % dice entender bien todas las solicitudes de requerimientos.

En prácticas ágiles de levantamiento de requerimientos, el equipo muestra un 32 % de conocimiento donde, a pesar de que el 71 % afirma conocer sobre el concepto de formato de historias de usuario, el 100 % considera que no se está utilizando ese formato y que no cuentan con criterios de aceptación en sus requerimientos.

Un 59 % de los recursos dice haber identificado un ID en el formato de los requerimientos usados actualmente.

Práctica Release management.

El nivel de madurez actual en prácticas de *release management* se encuentra en un 48 %, esto como resultado de los criterios definidos, los cuales obtuvieron un 55 % en el nivel de conocimiento de prácticas de *release management*, un 40 % en el nivel de estandarización de una metodología y un 50 % en el nivel de planificación, como se muestra en la tabla 10.

Tabla N.º 10

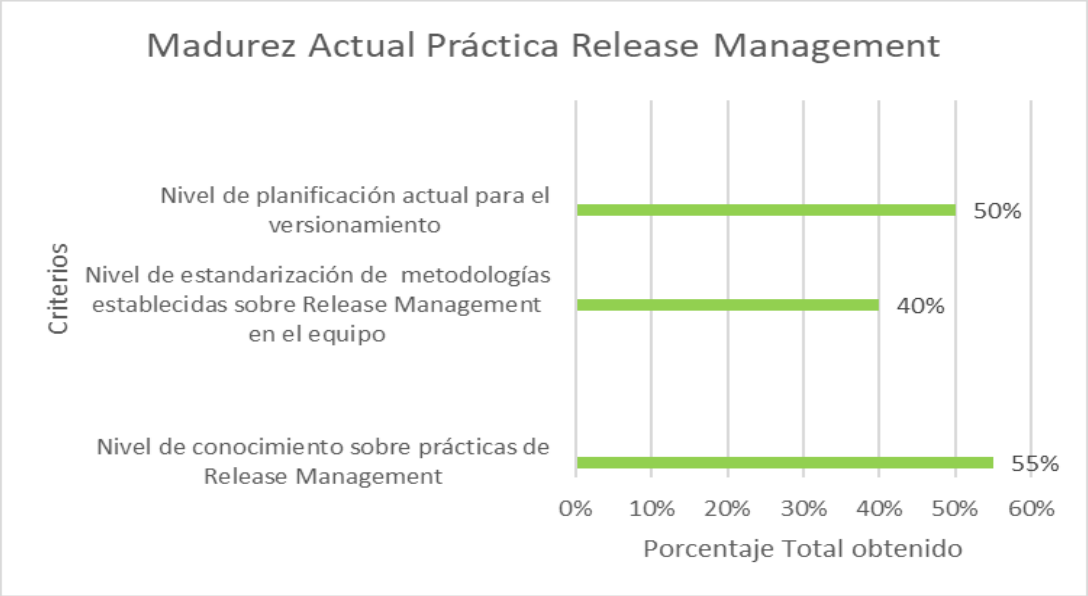
Madurez actual práctica de *Release management* equipo BA

Madurez Actual Práctica Release Management Equipo BA	
Nivel de conocimiento sobre prácticas de Release Management	55%
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	40%
Nivel de planificación actual para el versionamiento	50%

Nota: elaboración propia.

Figura N.º 22

Madurez actual práctica de *Release management* Equipo BA



Nota: elaboración propia.

En este equipo se visualiza un 55 % de nivel de conocimiento sobre prácticas ágiles, donde más del 50 % de los recursos manifiesta tener conocimientos básicos de gestión de versiones, elementos fundamentales de un repositorio y *software* GIT.

El nivel de estandarización de la práctica se encuentra en un 40 %, 47 % afirma que han manejado proyectos en repositorios, un 35 % manifiestan tener una herramienta para realizar los *releases* de sus proyectos y manejar estándares de nomenclatura para sus objetos. El 47 % de los recursos dicen contar con un estándar para la creación de repositorios, el 59 % considera que sí existe un control de notificaciones hacia el cliente cada vez que realizan un *release* y un 18 % ve inconvenientes para utilizar la metodología de *release*.

El equipo BA presenta un 50 % en el nivel de planificación que actualmente maneja, el 59 % dice tener una hora específica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio y un 41 % han realizado planificación de los *release* al inicio de un proyecto.

Práctica de aseguramiento y control de calidad.

El nivel actual de la práctica de aseguramiento y control de calidad se encuentra en un 40 %, se analizan los criterios y las respectivas variables, el resultado se muestra en la tabla 11.

Tabla N.º 11

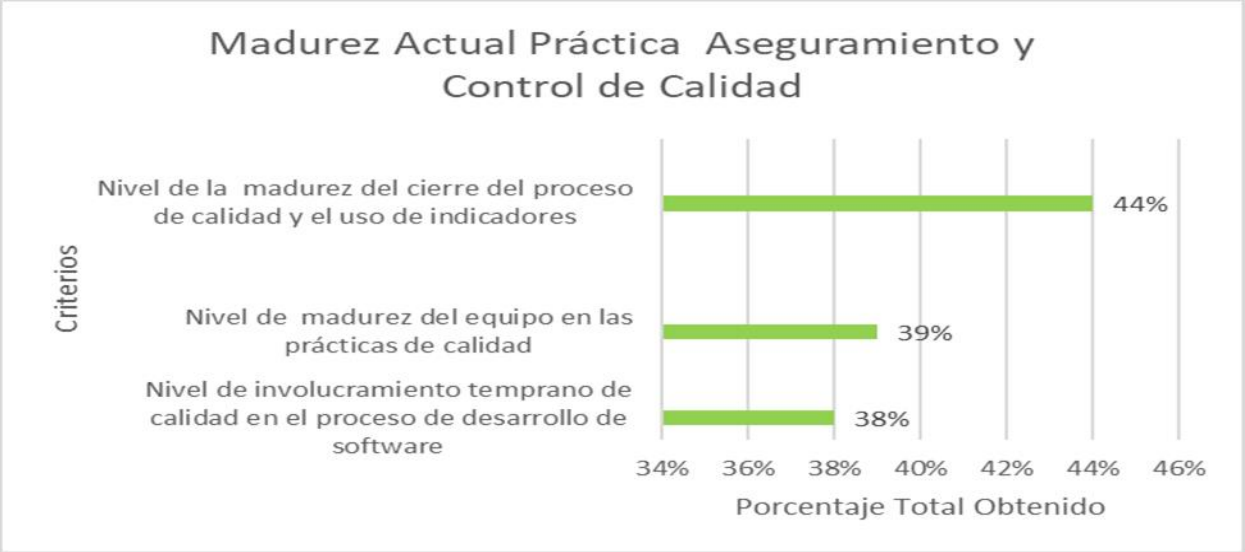
Madurez actual práctica de aseguramiento y control de calidad Equipo BA

Madurez actual práctica Aseguramiento y Control de Calidad Equipo BA	
Nivel de involucramiento temprano de calidad en el proceso de desarrollo de software	38%
Nivel de madurez del equipo en las prácticas de calidad	39%
Nivel de la madurez del cierre del proceso de calidad y el uso de indicadores	44%

Nota: elaboración propia.

Figura N.º 23

Madurez actual práctica de Aseguramiento y Control de Calidad Equipo BA



Nota: elaboración propia.

El criterio de involucramiento temprano de calidad en el proceso de desarrollo de *software* presenta un nivel de 38 %, este nivel se obtiene mediante las respuestas a las variables definidas para dicho criterio, donde el 67 % de los recursos considera que sí tienen un

involucramiento temprano de los ingenieros de calidad en sus proyectos, sin embargo, a pesar de existir este involucramiento, solo el 25 % de los recursos percibe que los ingenieros de calidad realizan un análisis de los requerimientos, un 42 % considera que los ingenieros de calidad cuentan con toda la información necesaria para desarrollar sus pruebas y solamente un 17 % creen que se logra la identificación en etapas tempranas de la necesidad de pruebas no funcionales (rendimiento, recuperación, etc.).

El nivel de madurez del equipo BA en las prácticas de calidad se posiciona en un 39%, donde se encuentra:

El 75 % considera que sí existe un proceso definido y estándar de desarrollo en el área de integración, el 50 % de los recursos considera que en su equipo sí existe un proceso definido y estándar de calidad y pruebas. Además, el 42 % de los recursos considera que sí se diseñan pruebas unitarias, sin embargo, solo el 25 % afirma que estas son ejecutadas y documentadas.

El 100 % del equipo manifiesta la ausencia de historias de usuario, así como de criterios de aceptación claros para el requerimiento. Por otra parte, el 67 % del equipo considera que sí se están documentando los cambios realizados por el cliente una vez iniciado el proyecto, pero solo el 50 % cree que se está especificando el impacto de los cambios solicitados por el cliente.

El 42 % del equipo considera que sí se está diseñando un plan de pruebas y casos de pruebas por proyecto.

La madurez del proceso de cierre de calidad y uso de kpis se encuentra en un 39 %, sus variables presentan el siguiente comportamiento:

Un 50% dice contar con nomenclatura para los objetos

Un 42 % del equipo confirman la documentación e información de los defectos utilizando una herramienta de seguimiento de defectos o reportes en documentos de office, sin embargo, un 33 % de los recursos conocen de la realización de monitoreos sobre la resolución de defectos y los esfuerzos necesarios para ello, el mismo porcentaje del equipo (33 %) realiza pruebas no funcionales en sus desarrollos de forma proactiva mientras un 83% las realiza de forma reactiva.

El 67 % considera que los resultados de las pruebas no funcionales son documentados e informados y el 33 % afirma que actualmente se documenta un informe final de calidad.

Y, únicamente, el 8 % utiliza kpis o indicadores, con el fin de contar con información para mejora continua.

Dados estos resultados, se puede percibir que las deficiencias que presenta el equipo IC pueden estarse presentando en los tres equipos, esta situación puede ser entendible perteneciendo los tres equipos a una misma área, compartiendo un rol de gestión bastante relevante. En el rol de líder técnico se percibe frustración y una oportunidad de mejora en la gestión importante.

No hay roles claros, el equipo está consciente de la existencia de algunas prácticas, pero las mismas no están estandarizadas para todos los tipos de desarrollos o trabajos, es decir, se tienen mapeadas algunas prácticas, pero no llegan a permear lo suficiente como para ser la práctica de todos.

Equipo Software Solutions (SO).

Práctica de levantamiento de requerimientos.

La práctica de levantamiento de requisitos en el equipo de SO presenta un nivel de un 44 %, según los criterios evaluados con el equipo, se obtiene de forma individual la puntuación mostrada en la tabla 12.

Tabla N.º 12

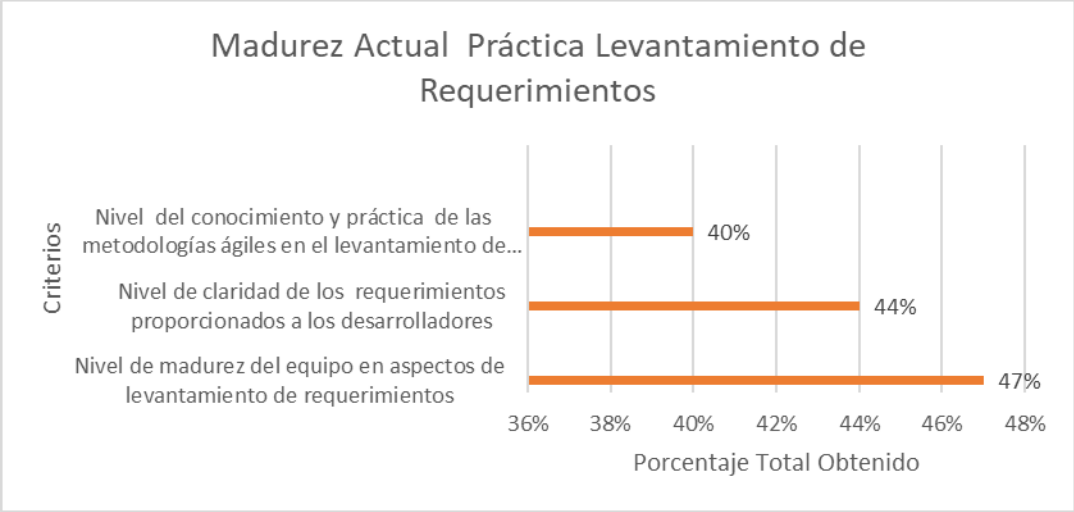
Madurez actual práctica Levantamiento de Requerimientos Equipo SO

Madurez Actual Levantamiento de Requerimientos Equipo SO	
Nivel de madurez del equipo en aspectos de levantamiento de requerimientos	47%
Nivel de claridad de los requerimientos proporcionados a los desarrolladores	44%
Nivel del conocimiento y práctica de las metodologías ágiles en el levantamiento de requerimientos.	40%

Nota: elaboración propia.

Figura N.º 24

Madurez actual práctica Levantamiento de requerimientos Equipo SO



Nota: elaboración propia.

El equipo SO presenta un nivel de madurez en aspectos de levantamiento de requerimientos de un 47 %, este criterio surge a raíz de los resultados de la evaluación de sus variables, donde el 100% de los recursos que conforman el equipo consideran que existe una etapa definida para el levantamiento de requerimientos, un 67 % afirma la realización de un análisis del problema antes de dar solución al requerimiento, un 17 % afirma que el recurso que levanta el requerimiento no desarrolla, mientras un 75 % afirma usar un mismo formato para el levantamiento de requerimientos en todos los proyectos.

Todo el equipo (100%) considera que no se está identificando un mínimo producto viable y solamente un 25 % confirma la presencia de método para documentar las dependencias entre los requerimientos.

El equipo considera que el nivel de claridad de los requerimientos es de un 44 %, el 75 % confirma que los requerimientos son analizados por el líder técnico antes de su desarrollo, solamente el 33 % manifiestan estar conformes con la información brindada en los requerimientos para desarrollar y únicamente un 25 % del equipo dice tener un buen entendimiento de todos los requerimientos que son proporcionados.

Según los datos proporcionados por el equipo, el conocimiento actual en prácticas ágiles de levantamiento de requerimientos se encuentra en un 40 %, donde el 50 % de los recursos dicen

conocer el formato de historias de usuario, sin embargo, a pesar de esto, el 100% de los recursos concuerdan con que actualmente este formato no se está utilizando, un 42 % afirma que los requerimientos sí cuentan con criterios de aceptación aunque no son llamados de esa manera y un 67 % afirma que los requerimientos actuales sí cuentan con un identificador.

Práctica Release management.

El equipo SO presenta una madurez del 55 % en lo que respecta a las prácticas de *release management*, los resultados de la evaluación según criterios y variables se presentan en la tabla 13.

Tabla N.º 13

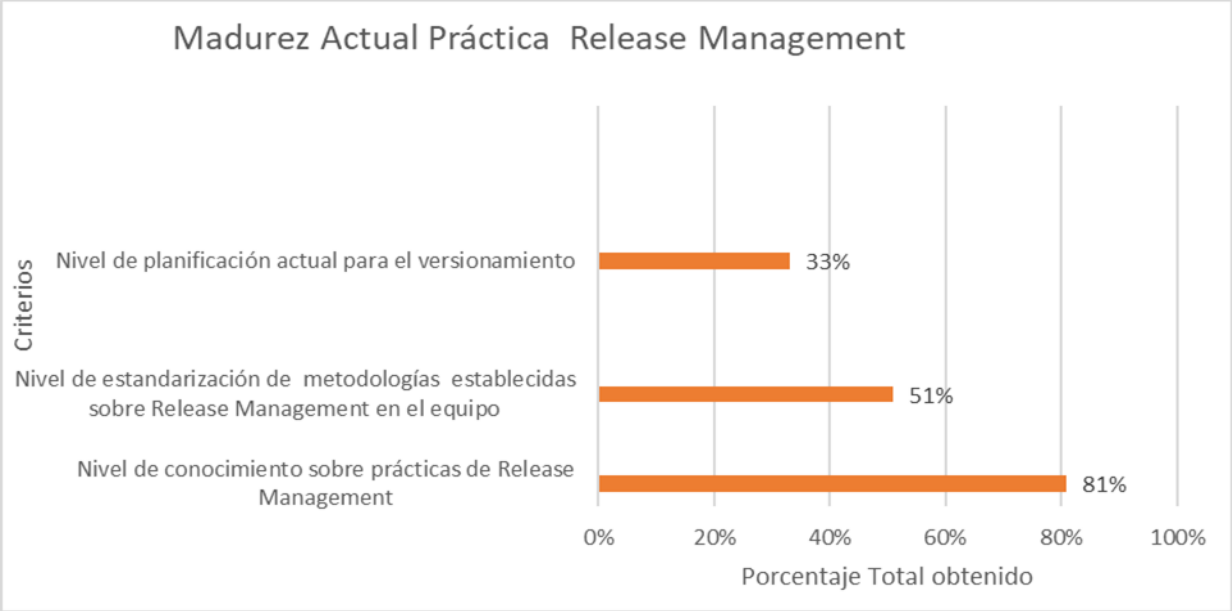
Madurez actual práctica de *Release management* Equipo SO

Madurez Actual Práctica Release Management Equipo SO	
Nivel de conocimiento sobre prácticas de Release Management	81%
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	51%
Nivel de planificación actual para el versionamiento	33%

Nota: elaboración propia.

Figura N.º 25

Madurez actual práctica *Release management* Equipo SO



Nota: elaboración propia.

En general, el equipo demuestra un conocimiento en estas prácticas de un 81 %, el 83 % del equipo considera que tienen los conocimientos básicos en las prácticas de *release*, así como de los elementos principales de un repositorio, un 75 % del equipo dice conocer la herramienta GIT.

Actualmente, el equipo SO presenta un nivel de estandarización del 60 %, un 42 % de los recursos cuentan con una herramienta para realizar los *release* de sus proyectos y un 50 % dice contar con estándares de nomenclatura para sus objetos.

El 67 % del equipo c afirma tener proyectos en repositorios, confirma la existencia de un estándar para la creación de repositorios y un control de notificaciones hacia el cliente cada vez que realizan un *release.* Solamente un 2 % ve inconveniente en implementar una metodología de *release management.*

Sobre la planeación de los *releases,* el equipo SO presenta un nivel de 33 %, donde el 42 % afirma tener una hora específica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio y solamente un 25 % afirma haber realizado una planeación de *releases* al inicio de los proyectos en los que ha participado.

Práctica de aseguramiento y control de calidad.

En el equipo SO se determina un nivel de madurez para la práctica de aseguramiento y control de calidad de un 29 %, para este equipo los criterios y las variables se comportaron de la siguiente manera mostrada en la tabla 14.

Tabla N.º 14

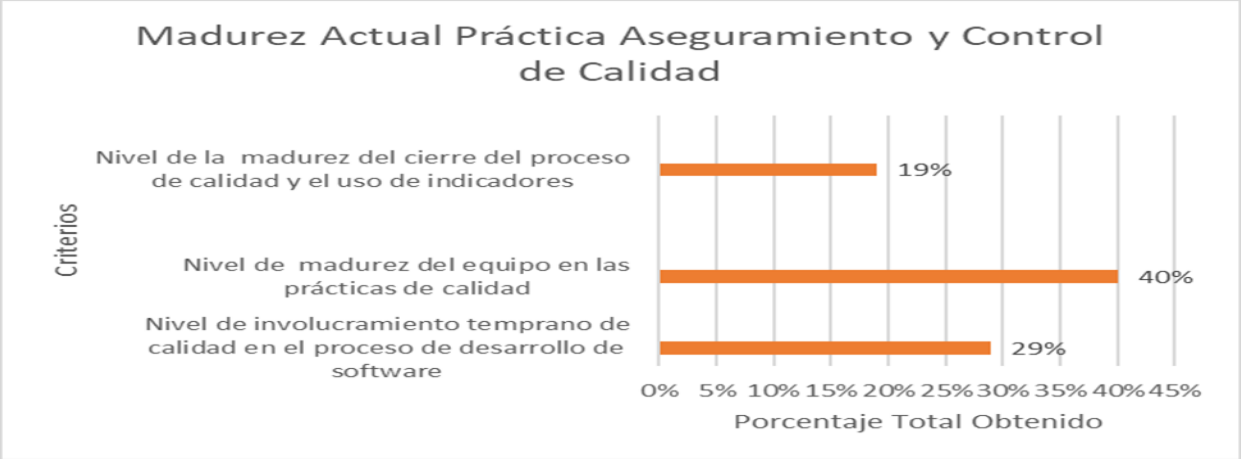
Madurez actual práctica de aseguramiento y control de calidad Equipo SO

Madurez Actual Práctica Ágil Aseguramiento y Control de Calidad Equipo SO	
Nivel de involucramiento temprano de calidad en el proceso de desarrollo de software	29%
Nivel de madurez del equipo en las prácticas de calidad	40%
Nivel de la madurez del cierre del proceso de calidad y el uso de indicadores	19%

Nota: elaboración propia.

Figura N.º 26

Madurez actual práctica de Aseguramiento y Control de Calidad Equipo BA



Nota: elaboración propia.

El equipo confirma un nivel de involucramiento de los ingenieros de calidad en etapas tempranas de un 42 %, el 33 % de los recursos concuerdan en que los requisitos son analizados por el ingeniero de calidad, con el objetivo de velar por los posibles riesgos, impedimentos, necesidades especiales o identificación de datos para pruebas, sin embargo, solo el 25 % de los recursos consideran que los recursos con rol de calidad cuentan con toda la información necesaria para desarrollar sus pruebas. Únicamente un 17 % considera que, actualmente, se identifica en etapas tempranas la necesidad de pruebas no funcionales (rendimiento, recuperación, etc.).

Se demuestra un nivel de madurez en prácticas de calidad de un 40 %, donde el 67 % de los recursos del equipo se encuentran de acuerdo sobre la existencia de un proceso definido y estandarizado de desarrollo en su unidad, un 42 % considera que existe un proceso definido con estándar de calidad en su unida, y un proceso definido de pruebas en sus equipos, además que se cuenta con criterios de aceptación para cada HU o RQ.

Un 33 % del equipo afirma haber contado con diseño de pruebas unitarias, sin embargo, solamente un 17 % confirma la ejecución de las mismas.

El 25 % de los recursos confirman la documentación de los resultados de las pruebas unitarias, la existencia de criterios de aceptación 100% claros, sin ambigüedades y alineados a asegurar el cumplimiento de la necesidad del negocio, el diseño de un plan de pruebas, así como el diseño de casos de pruebas en cada proyecto.

El 67 % de los recursos confirman la documentación de los cambios realizados por el cliente una vez iniciado el proyecto y la especificación del impacto de los cambios solicitados por el cliente. En el proceso de cierre de calidad, el equipo SO presenta una madurez del 34 %, conformado por los resultados de sus variables de la siguiente manera:

Un 17 % de los recursos confirman la existencia de un estándar de nomenclatura para los objetos creados.

Todo el equipo confirma que no se documentan e informan los defectos mediante una herramienta de seguimiento de defectos o reportes en documentos de office, por lo que no se cuenta con un monitoreo la resolución de defectos y los esfuerzos necesarios para ello, tampoco se realizan informes finales de calidad ni se utilizan actualmente Kpis o indicadores para la mejora continua.

El 42 % del equipo trata de realizar pruebas no funcionales en sus desarrollos de forma proactiva. Mientras el 67 % de los recursos manifiestan realizar pruebas no funcionales en sus desarrollos de forma reactiva. Solamente un 25 % considera que cuando se realizan las pruebas no funcionales los resultados son documentados y comunicados.

Si bien es cierto, el Equipo SO presenta el mayor nivel de madurez en las prácticas de levantamiento de requerimientos y *release management*, en la práctica de aseguramiento y calidad presenta un nivel mucho menor que el equipo IC y el equipo BA, esta diferencia es preocupante, ya que es esta práctica la que puede garantizar la mayor satisfacción del cliente sirviendo como filtro antes de la entrega de los desarrollos.

Al igual que los equipos anteriores, falta estandarización de los procesos y prácticas en general. Por medio del análisis de los resultados del conversatorio y los cuestionarios, se obtiene la claridad necesaria para identificar los principales retos de mejora que enfrenta el área, los mismos son utilizados para estudiar las posibles causas.

Diagrama Ishikawa (causa y efecto)

El objetivo de la realización del diagrama causa y efecto es identificar las causas del porqué el área en estudio considera necesario realizar una innovación y mejoras en la forma actual de trabajar, por lo que ese es el eje central sobre el cual se buscan y encuentran causas aparentes.

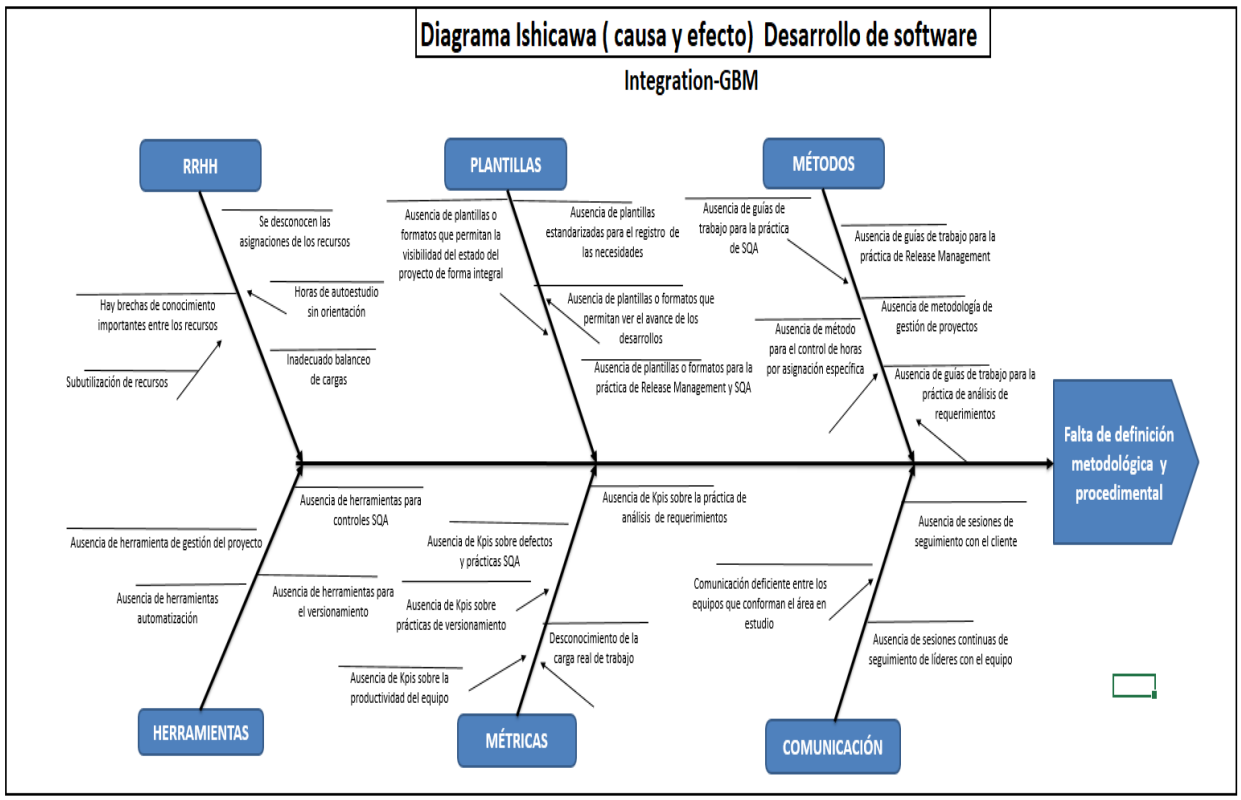
Con el fin de lograr la recolección de la mayor cantidad de datos, se siguió la siguiente metodología:

- Se realizan conversatorios con los líderes del área en estudio.
- Se explora el proceso actual y el sentir de los recursos que conforman los tres equipos.
- Se captura la información.
- Se organiza la información según corresponda en recursos humanos (RRHH), plantillas, herramientas, métodos, métricas y comunicación.

Como se puede ver en la siguiente figura 27, se identifican causas para los tres equipos en el área de recursos humanos, plantillas, métodos, herramientas, métricas y comunicación, las cuales están llevando al área a una necesidad de cambio, principalmente metodológico, procedimental y estandarizado.

Figura N.º 27

Diagrama de Ishikawa



Nota: elaboración propia.

Recursos humanos (RRHH).

Desconocimiento de las asignaciones de los recursos: actualmente el área de Integration está conformada por tres equipos, existen dos formas de gestionar los recursos: una es por parte el líder técnico y la otra es por medio de un puesto de planeación con el que cuenta la división de *software* de GBM. Actualmente, los líderes técnicos desconocen en qué están trabajando los recursos cuando la vía de gestión es por medio de los planners.

Horas de autoestudio sin orientación: se visualiza la posibilidad de contar con horas para el autoestudio, sin embargo, estas asignaciones no muestran tener una orientación clara que persiga los objetivos o las necesidades del área en estudio.

Subutilización de recursos: en este momento se evidencia la subutilización de los recursos catalogados como *Seniors* y que se desempeñan bajo el rol de líder técnico, esto ya que muchos

de sus recursos se encuentran gestionados por las *planners* o bien se encuentran con asignaciones de autoestudio.

Brechas de conocimiento importantes entre los recursos: actualmente los conocimientos de los recursos varían considerablemente, se cuenta con recursos muy capacitados y otros recursos muy *Junior*.

Inadecuado balanceo de cargas: las brechas de conocimiento están originando que existan recursos con altas cargas de trabajo, mientras otros recursos se encuentran flotantes a la espera de asignaciones.

Herramientas.

Ausencia de herramienta de gestión del proyecto: no se cuenta actualmente con una herramienta que logre organizar y dar visibilidad de las diferentes prácticas que forman parte del proyecto como los requerimientos del cliente, las tareas en que se encuentra cada recurso, fechas de entrega, riesgos, lecciones aprendidas; parte de la información la manejan los líderes técnicos, otra información la manejan los *planners.*

Ausencia de herramientas automatización: actualmente, el área de integración no ha invertido en herramientas que permitan automatizar los procesos.

Ausencia de herramientas para controles de aseguramiento y control de calidad: no se cuenta con procedimientos ni prácticas de control y aseguramiento de la calidad de forma estandarizada, no existe documentación que soporten los resultados de las prácticas actuales en todos los trabajos que se realizan.

Ausencia de herramientas para el versionamiento: no todos los equipos implementan las prácticas de *release management,* el único equipo que lo práctica se encuentra en etapas muy tempranas por lo que la madurez en esta práctica al día de hoy es muy baja, solamente un equipo de tres cuenta con la herramienta necesaria.

Plantillas.

Ausencia de plantillas o formatos que permitan la visibilidad del estado del proyecto de forma integral.

Ausencia de plantillas estandarizadas para el registro de las necesidades: no se cuenta con un formato estándar para el registro de los requerimientos que permita documentar la información necesaria de forma clara y precisa para poder desarrollar la solución.

Ausencia de plantillas o formatos que permitan ver el avance de los desarrollos: no hay visibilidad en ninguna plantilla o herramienta de los avances en los desarrollos.

Ausencia de plantillas o formatos para la práctica de *release management* y SQA: no existe documentación sobre los resultados actuales de las prácticas de control de la calidad y versionamiento que se están realizando.

Métricas.

Ausencia de kpis (*key performance indicator,* conocido también como indicador clave o medidor de desempeño) sobre defectos y prácticas SQA, versionamiento, productividad, requerimientos: actualmente no existen métricas de referencia de cada desarrollo realizado sobre las diferentes prácticas que conforman el desarrollo de *software*.

Desconocimiento de la carga real de trabajo: el no conocer la carga real de trabajo impide realizar planeaciones estratégicas y pronósticos de trabajos futuros, por lo que está limitando la posibilidad de ofertar nuevos trabajos.

Métodos.

Ausencia de guías de trabajo para la práctica de SQA, *release management* y requerimientos: si bien es cierto los equipos están realizando pruebas a los desarrollos, sin embargo, no existe una guía de trabajo estándar para toda el área, igual sucede con la práctica de *release management,* donde solo un equipo de tres está iniciando su implementación y con la práctica de requerimientos donde tanto la forma de levantamiento como la documentación varía de un equipo a otro y en algunas ocasiones de un cliente a otro.

Ausencia de método para el control de horas por asignación específica: se desconoce realmente cuántas horas se están invirtiendo en una tarea en específico, lo que puede llevar a los equipos a originar desperdicios de tiempo innecesarios.

Ausencia de metodología de gestión de proyectos: si bien GBM a lo largo de los años ha desarrollado sus proyectos bajo la metodología tradicional de gestión de proyectos, esta no ha

permeado en el área de Integration, por la naturaleza de sus desarrollos o por falta de definición, la misma no se está implementando de forma estandarizada.

Comunicación.

En general, se evidencia una falta de comunicación entre los líderes técnicos y los *planners/Project managers,* entre los líderes técnicos y sus equipos, entre los equipos como tal y entre GBM y los clientes. Se percibe frustración y agotamiento en algunos de los recursos entrevistados.

Análisis de fortalezas y debilidades

Ya conocidas las causas y validadas con el equipo, se considera importante analizar las fortalezas y debilidades desde el punto de vista de otros recursos que trabajan indirectamente con el área de Integración sin dejar por fuera la representación del equipo. Para este fin, se utiliza el análisis de las fortalezas y debilidades del área, la tabla multivoto y la matriz EFI, a continuación, se presentan los resultados obtenidos.

Tabla multivoto.

La tabla multivoto permite calificar los problemas, las causas de los problemas o las limitaciones de un proceso de mejoramiento continuo.

Para el presente estudio, una vez evidenciadas las oportunidades de mejora, se decide analizar a lo interno la situación actual, tomando en cuenta las debilidades identificadas y las posibles causas encontradas en el diagrama de Ishikawa, las cuales en apariencia están impidiendo un mejor rendimiento de los equipos que conforman el área de Integration, estas debilidades y causas son analizadas, votadas y priorizadas mediante la tabla multivoto.

De la misma manera, se toman las fortalezas identificadas y se priorizan por votación, así como también se determina el ponderado que poseen ambos criterios (Fortalezas, Debilidades /causas) a nivel del área de Integration. Para realizar la votación, se seleccionan 10 recursos con roles importantes dentro del área que hoy en día son contundentes en la toma de decisiones y gestionamiento de los desarrollos que se trabajan, estos 10 roles son los siguientes:

-Líder técnico (3)

-Planners/Project managers (3)

-Procesos (1)

-Arquitectos de soluciones (1)

-Desarrolladores (1)

-Ingeniero de calidad (1)

A continuación, se presentan los resultados de la votación, donde 1 representa lo más importante o más crítico y 5 significa lo menos importante o crítico:

Tabla N.º 15

Tabla Multivoto

FACTORES	CALIFICACIÓN				
Fortalezas	**1**	**2**	**3**	**4**	**5**
Existe una cantidad considerable de recursos que permite reestructurar la organización de los equipos y el área en sí.		X	XXXX	XXX	XX
Existe diversidad de conocimiento entre los recursos, lo que permite una buena diversidad en los tipos de productos atendidos.	XXX	XX	XXXX	X	
Las destrezas y habilidades de los recursos son conocidas por los líderes y los *planners.*		X	XXXX	XXX	XX
Existe la apertura para implementar cambios y mejoras	XXXX	XX	XXXX	X	X
Hay tres recursos definidos bajo el rol de líderes técnicos dentro del área de Integration.		X	XXXX	XXX	XX
Existe comunicación con el cliente.	XX	XXX	XXXXX		
Se cuenta con ambientes de desarrollo y calidad.	XXXX	XX	XXXX		
Se contabilizan tiempos de espera por atrasos del cliente.		X	XXXXX	XXX	X
Existe la asignación de tiempo para autoestudio para los recursos.	XXX	XX	XXX	X	X
Debilidades/ Causas	**1**	**2**	**3**	**4**	**5**
No se cuenta con herramientas de gestión efectivas.	XXXXX	XXX	XX		
No se utilizan métricas para ninguno de los procesos.	XXXXX	XXXX	XXX		
No se está siguiendo una metodología como tal ni de desarrollo ni de gestión de proyectos.	XXXX	XXXX	XX		
Se desconoce la capacidad real de producción.	XX	XXXX	XX	XX	
No se cuenta con roles definidos de forma clara.	XXX	XXXX	XX	X	
No se cuenta con procesos estandarizados.	XXX	XXXXX	X	X	
Falta de *coaching* en temas de gestión, optimización de la producción, balanceo de cargas			XXX	XXXX	XXX
No hay estandarización en el uso de plantillas o artefactos.	XX	XXXX	XXXX	X	
No realizan ningún tipo de ceremonia o sesiones de seguimiento entre líderes y equipo.	XX	XXX	XXXX	X	
Horas de autoestudio sin orientación.		X	X	XXXX	XXXX

Nota: elaboración propia.

Una vez realizada la votación, se procede a realizar el conteo, los cuales se comportaron de la siguiente manera:

Tabla N.º 16

Resumen de votación de todos los miembros

FACTORES	CALIFICACIÓN				
Fortalezas	**1**	**2**	**3**	**4**	**5**
Existe una cantidad considerable de recursos que permite reestructurar la organización de los equipos y el área en sí.	**0**	**1**	**4**	**3**	**2**
Existe diversidad de conocimiento entre los recursos lo que permite una buena diversidad en los tipos de productos atendidos.	**3**	**2**	**4**	**1**	
Las destrezas y habilidades de los recursos son conocidas por los líderes y los *planners.*		**1**	**4**	**3**	**2**
Existe la apertura para implementar cambios y mejoras	**4**	**2**	**4**	**1**	**1**
Hay tres recursos definidos bajo el rol de líderes técnicos dentro del área de integración.		**1**	**4**	**3**	**2**
Existe comunicación con el cliente.	**2**	**3**	**5**		
Se cuenta con ambientes de desarrollo y calidad.	**4**	**2**	**4**		
Se contabilizan tiempos de espera por atrasos del cliente.		**1**	**5**	**3**	**1**
Existe la asignación de tiempo para autoestudio para los recursos.	**3**	**2**	**3**	**1**	**1**
Debilidades/ Causas	**1**	**2**	**3**	**4**	**5**
No se cuenta con herramientas de gestión efectivas.	**5**	**3**	**2**		
No se utilizan métricas para ninguno de los procesos.	**5**	**4**	**3**		
No se está siguiendo una metodología como tal ni de desarrollo ni de gestión de proyectos.	**4**	**4**	**2**		
Se desconoce la capacidad real de producción.	**2**	**4**	**2**	**2**	
No se cuenta con roles definidos de forma clara.	**3**	**4**	**2**	**1**	
No se cuenta con procesos estandarizados.	**3**	**5**	**1**	**1**	
Falta de *coaching* en temas de gestión, optimización de la producción, balanceo de cargas.			**3**	**4**	**3**
No hay estandarización en el uso de plantillas o artefactos.	**2**	**4**	**4**	**1**	
No realizan ningún tipo de ceremonia o sesiones de seguimiento entre líderes y equipo.	**2**	**3**	**4**	**1**	
Horas de autoestudio sin orientación.		**1**	**1**	**4**	**4**

Nota: elaboración propia.

Seguidamente, multiplicando cada factor por la cantidad de votaciones encontradas según cada punto, se obtienen los totales representados en la tabla 17.

Tabla N.º 17

Votos totalizados

FACTORES	CALIFICACIÓN					TOTAL
Fortalezas	**1**	**2**	**3**	**4**	**5**	
Existe una cantidad considerable de recursos que permite reestructurar la organización de los equipos y el área en sí.	**0**	**2**	**12**	**12**	**10**	**36**
Existe diversidad de conocimiento entre los recursos lo que permite una buena diversidad en los tipos de productos atendidos.	**3**	**2**	**12**	**4**	**0**	**21**
Las destrezas habilidades de los recursos son conocidas por los líderes y los *planners.*	**0**	**2**	**12**	**12**	**10**	**36**
Existe la apertura para implementar cambios y mejoras.	**4**	**4**	**12**	**4**	**5**	**29**
Hay tres recursos definidos bajo el rol de líderes técnicos dentro del área de integración.	**0**	**2**	**12**	**12**	**10**	**36**
Existe comunicación con el cliente.	**2**	**6**	**15**	**0**	**0**	**23**
Se cuenta con ambientes de desarrollo y calidad.	**4**	**4**	**9**	**12**	**5**	**34**
Se contabilizan tiempos de espera por atrasos del cliente.	**0**	**2**	**15**	**12**	**5**	**34**
Existe la asignación de tiempo para autoestudio para los recursos.	**3**	**4**	**12**	**4**	**5**	**28**
Debilidades	**1**	**2**	**3**	**4**	**5**	**TOTAL**
No se cuenta con herramientas de gestión efectivas.	**5**	**6**	**6**	**0**	**0**	**17**
No se utilizan métricas para ninguno de los procesos.	**5**	**8**	**9**	**0**	**0**	**22**
No se está siguiendo una metodología como tal ni de desarrollo ni de gestión de proyectos.	**4**	**8**	**6**	**0**	**0**	**18**
Se desconoce la capacidad real de producción.	**2**	**8**	**6**	**8**	**0**	**24**
No se cuenta con roles definidos de forma clara.	**3**	**8**	**6**	**4**	**0**	**21**
No se cuenta con procesos estandarizados.	**3**	**10**	**3**	**4**	**0**	**20**
Falta de *coaching* en temas de gestión, optimización de la producción, balanceo de cargas.	**0**	**0**	**9**	**16**	**16**	**41**
No hay estandarización en el uso de plantillas o artefactos.	**2**	**8**	**12**	**4**	**0**	**26**
No realizan ningún tipo de ceremonia o sesiones de seguimiento entre líderes y equipo.	**2**	**6**	**12**	**4**	**0**	**24**
Horas de autoestudio sin orientación.	**0**	**2**	**3**	**16**	**20**	**41**

Nota: elaboración propia.

Mediante este método, se logra priorizar los factores internos según importancia y criticidad para el área de Integration, de forma tal que el criterio con puntaje menor es prioritario, quedando de la siguiente manera:

Fortalezas:

- Existe diversidad de conocimiento entre los recursos, lo que permite una buena diversidad en los tipos de productos atendidos.
- Existe comunicación con el cliente.
- Existe la asignación de tiempo para autoestudio para los recursos.
- Existe la apertura para implementar cambios y mejoras.
- Se cuenta con ambientes de desarrollo y calidad.
- Se contabilizan tiempos de espera por atrasos del cliente.
- Existe una cantidad considerable de recursos que permite reestructurar la organización de los equipos y el área en sí.
- Las destrezas y habilidades de los recursos son conocidas por los líderes y los *planners.*
- Hay tres recursos definidos bajo el rol de líderes técnicos dentro del área de integración.

Debilidades

- No se cuenta con herramientas de gestión efectivas.
- No se está siguiendo una metodología como tal ni de desarrollo ni de gestión de proyectos.
- No se cuenta con procesos estandarizados.
- No se cuenta con roles definidos de forma clara.
- No se utilizan métricas para ninguno de los procesos.
- Se desconoce la capacidad real de producción.
- No realizan ningún tipo de ceremonia o sesiones de seguimiento entre líderes y equipo.

- No hay estandarización en el uso de plantillas o artefactos.
- Falta de *coaching* en temas de gestión, optimización de la producción, balanceo de cargas.
- Horas de autoestudio sin orientación.

Para determinar la ponderación, se tomaron en cuenta los niveles de importancia de 1 y 2, dado que representan la mayor importancia o criticidad para el área en estudio, por lo tanto, al contar con 10 recursos para la votación, se divide la suma de los votos de la importancia 1 y 2 entre el total, el cual es 10. Ese resultado es una ponderación individual, lo que conlleva a sumar todas las ponderaciones individuales para determinar la total (suma sea igual a 1). Por lo tanto, la ponderación total se divide en cada ponderación individual correspondiente y así se determina cuÁl es el peso de cada factor, obteniendo así la priorización y su peso para ser utilizado en la matriz EFI.

En la tabla No 18 se pueden observar los resultados de dichas ponderaciones:

Tabla N.º 18

Ponderaciones por Factor

FACTORES INTERNOS	PONDERACIÓN			
Fortalezas	**Votos**	**N**	**Ponderación Individual**	**Peso**
Existe una cantidad considerable de recursos que permite reestructurar la organización de los equipos y el área en sí.	**1**	**10**	**0,1**	**0,01**
Existe diversidad de conocimiento entre los recursos lo que permite una buena diversidad en los tipos de productos atendidos.	**5**	**10**	**0,5**	**0,06**
Las destrezas y habilidades de los recursos son conocidas por los líderes y los *planners.*	**1**	**10**	**0,1**	**0,01**
Existe la apertura para implementar cambios y mejoras.	**6**	**10**	**0,6**	**0,07**
Hay tres recursos definidos bajo el rol de líderes técnicos dentro del área de integración.	**1**	**10**	**0,1**	**0,01**
Existe comunicación con el cliente.	**5**	**10**	**0,5**	**0,06**
Se cuenta con ambientes de desarrollo y calidad.	**6**	**10**	**0,6**	**0,07**
Se contabilizan tiempos de espera por atrasos del cliente.	**1**	**10**	**0,1**	**0,01**
Existe la asignación de tiempo para autoestudio para los recursos.	**5**	**10**	**0,5**	**0,06**

Debilidades/ Causas				
No se cuenta con herramientas de gestión efectivas.	**8**	**10**	**0,8**	**0,09**
No se utilizan métricas para ninguno de los procesos.	**9**	**10**	**0,9**	**0,10**
No se está siguiendo una metodología como tal ni de desarrollo ni de gestión de proyectos.	**8**	**10**	**0,8**	**0,09**
Se desconoce la capacidad real de producción.	**6**	**10**	**0,6**	**0,07**
No se cuenta con roles definidos de forma clara.	**7**	**10**	**0,7**	**0,08**
No se cuenta con procesos estandarizados.	**9**	**10**	**0,9**	**0,10**
Falta de *coaching* en temas de gestión, optimización de la producción, balanceo de cargas.	**0**	**10**	**0**	**0,00**
No hay estandarización en el uso de plantillas o artefactos.	**6**	**10**	**0,6**	**0,07**
No realizan ningún tipo de ceremonia o sesiones de seguimiento entre líderes y equipo.	**5**	**10**	**0,5**	**0,06**
Horas de autoestudio sin orientación.	**1**	**10**	**0,1**	**0,01**
TOTAL			**9**	**1**

Nota: elaboración propia.

Matriz EFI.

Una vez que se priorizan los factores internos con su peso, se utiliza la Matriz EFI para conocer el balance entre las fortalezas y las debilidades presentes al día de hoy en el área de estudio. Se realiza el respectivo cálculo de cada factor con su peso ponderado y su calificación acorde con las prioridades que dieron resultado de la tabla multivoto. En la tabla nro. 19, se puede observar el resultado de la matriz EFI y sus respectivos cálculos.

Tabla N.º 19

Matriz EFI

Factores Internos Clave		Importancia Ponderación	Clasificación Evaluación	Valor
Fortalezas				
1.	Existe una cantidad considerable de recursos que permite reestructurar la organización de los equipos y el área en sí.	1%	**1**	0,01
2.	Existe diversidad de conocimiento entre los recursos lo que permite una buena diversidad en los tipos de productos atendidos.	6%	1	0,06

3.	Las destrezas y habilidades de los recursos son conocidas por los líderes y los *planners.*	1%	3	0,03
4.	Existe la apertura para implementar cambios y mejoras.	7%	3	0,20
5.	Hay tres recursos definidos bajo el rol de líderes técnicos dentro del área de integración.	1%	4	0,04
6.	Existe comunicación con el cliente.	6%	4	0,22
7.	Se cuenta con ambientes de desarrollo y calidad.	7%	5	0,33
8.	Se contabilizan tiempos de espera por atrasos del cliente.	1%	5	0,06
9.	Existe la asignación de tiempo para autoestudio para los recursos.	6%	5	0,28
Subtotal				1,23333
Debilidades				
1.	No se cuenta con herramientas de gestión efectivas.	9%	1	0,09
	Factores internos clave	Importancia calificación	Clasificación evaluación	valor
2.	No se utilizan métricas para ninguno de los procesos.	10%	1	0,10
3.	No se está siguiendo una metodología como tal ni de desarrollo ni de gestión de proyectos.	9%	2	0,18
4.	Se desconoce la capacidad real de producción.	7%	2	0,13
5.	No se cuenta con roles definidos de forma clara.	8%	2	0,16
6.	No se cuenta con procesos estandarizados.	10%	3	0,30
7.	Falta de *coaching* en temas de gestión, optimización de la producción, balanceo de cargas.	0%	3	0,00
8.	No hay estandarización en el uso de plantillas o artefactos.	7%	4	0,27
9.	No realizan ningún tipo de ceremonia o sesiones de seguimiento entre líderes y equipo.	6%	5	0,28
10	Horas de autoestudio sin orientación.	1%	5	0,06
Sub Total				1,56

Total	**100%**		**1**

Nota: elaboración propia.

En la tabla 19, se muestra la matriz con los factores, el peso ponderado y su calificación de acuerdo con su debilidad o fortaleza y el valor ponderado obtenido, con el fin de conocer el balance actual entre las fortalezas y debilidades que presenta. Como se muestra en el cuadro, se tiene un resultado igual a 1, lo que refleja que a lo interno el área de Integration se encuentra muy por debajo de la media (2,5). Esto da a entender que internamente no se cuenta con un balance adecuado entre las fortalezas y las debilidades, pesando más las debilidades que las fortalezas.

CAPÍTULO V. CONCLUSIONES Y RECOMENDACIONES

- Los equipos suelen ser gestionados por un proveedor externo a GBM, el líder técnico GBM se involucra cuando es requerido en temas de gestión únicamente, lo que está dejando sin control a GBM de situaciones que afectan a sus clientes.
- Es necesaria una reestructuración de los procesos de construcción, gestión y mejora continua.
- Se requiere de una reestructuración metodológica, procedimental y de roles, que pueda ser aplicada a los tres equipos del área.
- Se requiere desarrollar prácticas ágiles de desarrollo.
- Es necesaria la creación de un proceso de gestión y gobierno de requerimientos que asegure la entrada de insumos de calidad al proceso de construcción y automatización, ya que de momento no hay forma de automatizar las integraciones que se tienen.
- Es necesaria la definición de la estructura orgánica orientada a fábrica y la implementación de metodologías acordes con esta estrategia.
- Es necesario establecer mecanismos para estandarización, optimización de estimaciones y control de calidad desde la construcción.
- Es indispensable la creación de métricas para control del desperdicio (tiempos muertos y *reworking*).
- Es necesario un análisis/ asignación de roles y optimización de recursos.
- Se evidencia una subutilización del líder técnico, al tener únicamente cuatro recursos asignados, este trabajo puede ser realizado por un Scrum Máster y un *product owner* (Roles ágiles).
- Se debe desarrollar una capacidad de producción que permita que los *planners* no sobre carguen algunos recursos, mientras otros están más ociosos. Esto se alinea con el proceso de planes de inducción claramente definidos basados en capacidad instalada.

Como recomendaciones que quedan fuera del alcance del presente estudio por temas de tiempo y enfoque, se identifican las siguientes:

- Se requiere una revisión y probablemente ajustes del rol del Customer Success.

- Se debe dar *coaching* en temas de gestión de costos, optimización de la producción y balanceo de cargas de trabajo.
- Se requiere desarrollar procesos de *mentoring*, *coaching* y desarrollo del nuevo personal, por parte de los ingenieros más *Senior.*
- Definir planes de capacitación holísticos y multiplataformas (IBM-OTROS).

CAPÍTULO VI PROPUESTA

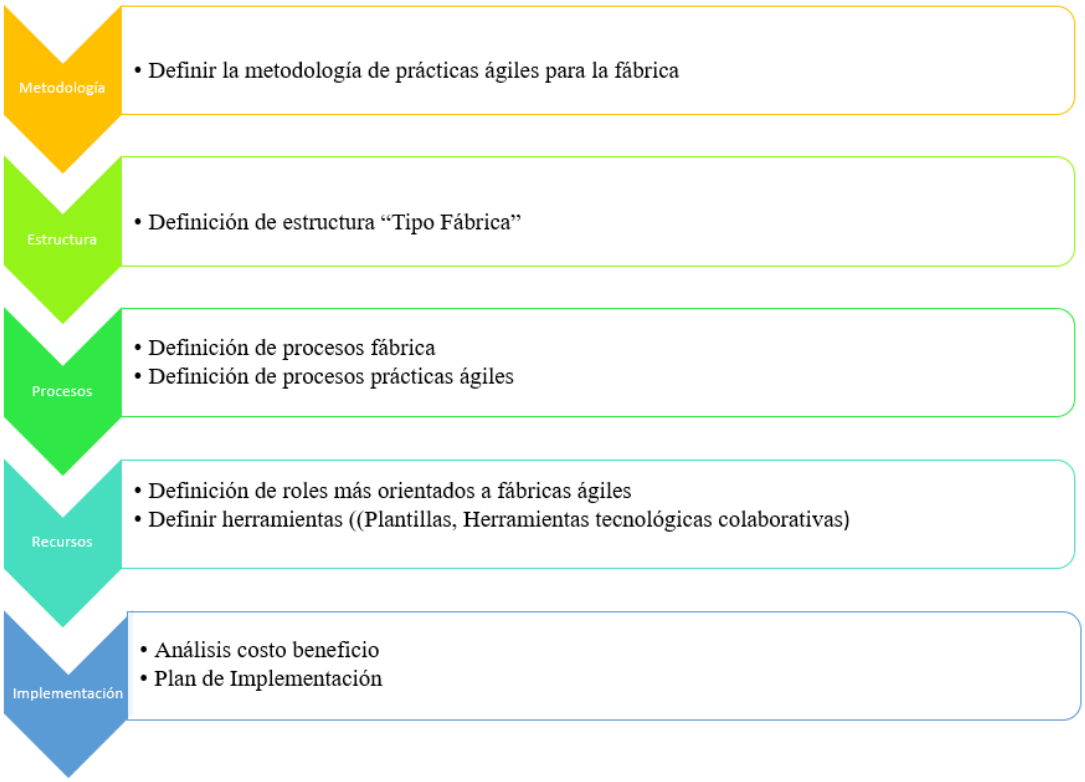
Según la necesidad de innovación presentada por el área en estudio y los resultados del diagnóstico realizado, la presente propuesta se basa en el diseño de una metodología, la cual busca integrar prácticas ágiles de desarrollo de *software*, dinámicas de trabajo y roles que junto a una serie de herramientas tecnológicas colaborativas brinden la flexibilidad necesaria y la visibilidad requerida en tiempo real para soportar un modelo de cambio continuo.

Para establecer la metodología y según las necesidades evidenciadas en el diagnóstico, se propone a su vez un reacomodo de los equipos, de forma tal que se logre una estructura en células dedicadas a productos específicos “tipo fábrica”, los cuales logren soportar la metodología y aprovechar al máximo la cantidad de recursos y las habilidades con las que cuenta hoy en día el área de Integration.

Las prácticas ágiles de *software*, como su nombre lo indica, giran en torno a la agilidad donde estén presentes entregas continuas de valor a negocio, por lo tanto, es muy importante que las industrias de tecnología entiendan y compartan un modelo que represente la necesidad del negocio. Entendiendo esta necesidad, la propuesta de la solución estará conformada por los siguientes apartados:

Figura N.º 28

Apartados de la propuesta



Nota: elaboración propia.

Propuesta metodológica

Para las células que conforman la fábrica de desarrollo del área de Integration, se propone la metodología Scrumban (combinación de marco Scrum y Kanban), las cuales han sido explicadas ampliamente en el marco teórico de la presente investigación.

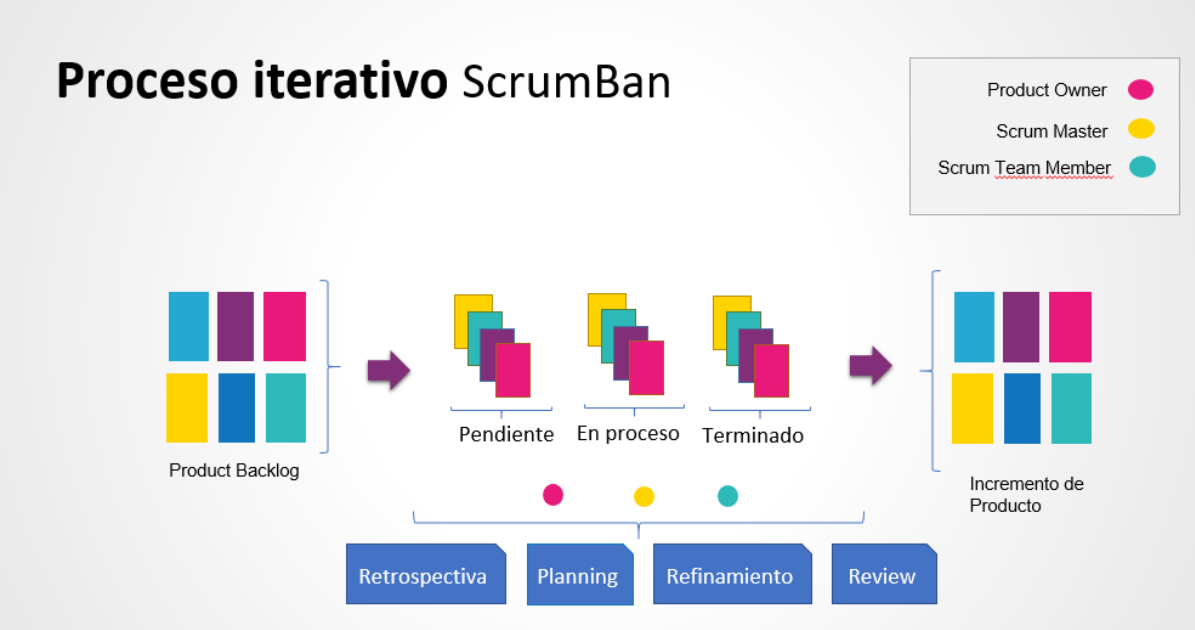
Scrumban

Se propone la utilización del marco Scrumban para el desarrollo de requerimientos de las célula, debido a la naturaleza y duración de los trabajos que asume el área en estudio. Scrumban es una metodología que establece un método visual y en el cual el trabajo en curso está limitado. El objetivo es crear un flujo constante y ágil de atención de necesidades para minimizar el tiempo de entrega; al haber un flujo constante, no se ejecutan *sprint*s (marcos de tiempo fijos) como en Scrum, pero sí se tienen ciclos de tiempo establecidos. Para el área de Integration, se proponen ciclos de dos semanas, con el fin de propiciar la inspección y adaptación de los equipos analizando métricas y haciendo retrospectivas.

Se determina que, al salir un elemento terminado de cada una de las etapas del flujo de trabajo de los equipos solucionadores, entrará un nuevo elemento para ser atendido de acuerdo a la prioridad establecida. El principio general de Scrumban es que un elemento sale, y un elemento entra. Para garantizar este principio, se propone la utilización de un tablero Kanban que tiene definido claramente el límite de elementos que están en “trabajo en progreso” o WIP, con el fin de que el equipo no se desenfoque de las prioridades. Las necesidades son dadas según la demanda de urgencias o incidentes presentados al IMT(Integration Management Team) para resolver.

Figura N.º 29

Diagrama de Proceso Scrumban



Nota: elaboración propia.

Propuesta de estructura

Se propone la definición de una estructura tipo fábrica, donde cada célula maneja líneas de producción especializadas en uno o varios productos de los que actualmente trabajan los tres equipos estudiados. Además, se propone una línea denominada Trainees, donde estarán todos aquellos recursos de primer ingreso que se encuentran en capacitación, o bien, aquellos recursos flotantes que no han sido asignados a clientes y deben preparar certificaciones según el plan de crecimiento de la empresa.

Se recomienda la implementación de una fábrica liderada por un IMT (Integration Management Team) representado por un Scrum Coach, un Program Manager, un Líder técnico, un analista y un arquitecto empresarial, quienes gestionarán una estructura de cinco células. Estas células serán lideradas, a su vez, por un OMT (Operation Management Team) representado por líderes con roles de calidad, gestión ágil, desarrollo y análisis de requerimientos, acompañados siempre por el *product owner* de las soluciones. Dichos roles se presentarán en los siguientes apartados de la presente propuesta.

Las células se constituyen por dos carriles, el primer carril funciona como línea de desarrollo de proyectos o requerimientos, donde entrarán todos los requerimientos aprobados para desarrollar por los equipos solucionadores y un Fast Line (carril rápido), donde se atenderán requerimientos o solicitudes urgentes. Para poder determinar cuáles solicitudes pueden pasar por el Fast Line y cuáles se deben atender primero en ambos carriles, se propone el uso de un modelo de priorización.

La fábrica debe contar con un *backlog* (Listado) de necesidades principal como resultado del proceso de recepción de solicitudes (provenientes directamente de clientes con bolsas de horas previamente adquiridas o provenientes del área de ventas como nuevos proyectos/ requerimientos por hacer). Estas necesidades previamente son revisadas y aprobadas por el IMT, quien es el responsable de la asignación a la célula correspondiente. Una vez ingresada a la célula, el OMT realiza una revisión y aprobación correspondiente, esta aprobación logra el ingreso del requerimiento al *backlog* de necesidades propio de la célula y se realiza según se presentará más adelante como parte de las prácticas ágiles.

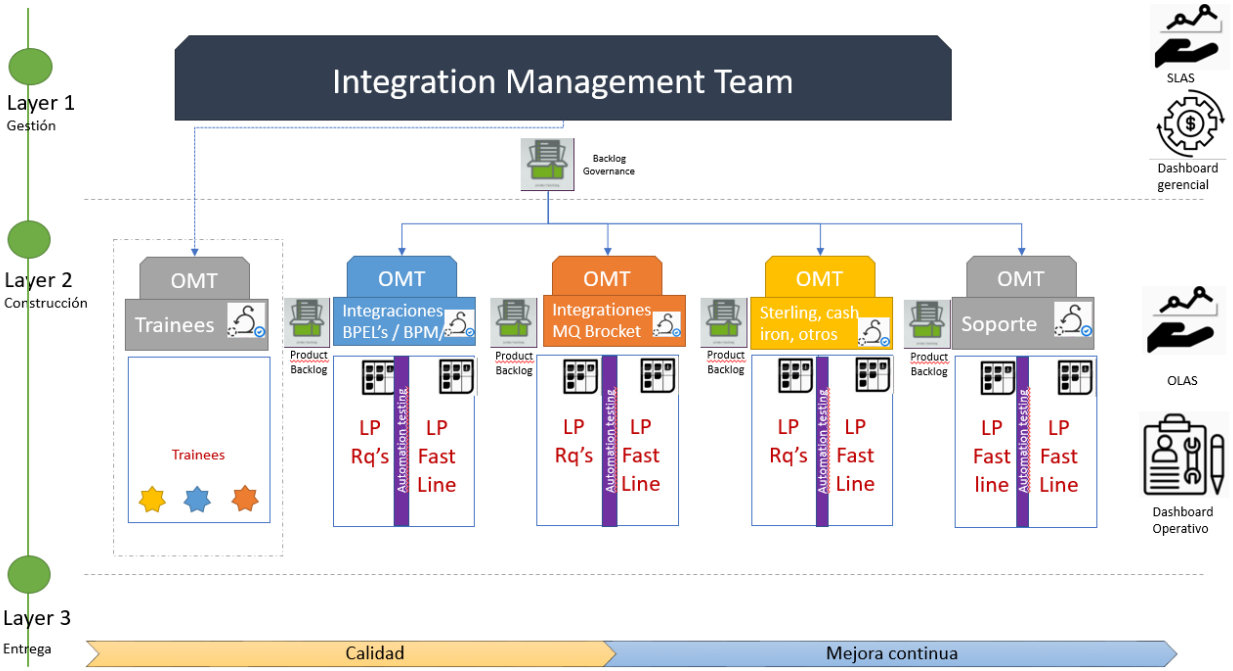
Los procesos de calidad y mejora continua se mantienen a lo largo de toda la estructura cubriendo así las necesidades de las células para garantizar la calidad de los trabajos que se realicen.

Como parte de la estrategia, será necesario definir Acuerdos a nivel de servicio (SLAS), Acuerdos a nivel operativo (OLAS) y Dashboard Gerenciales, sin embargo, por razones de tiempo, esto no se encuentra dentro del alcance de la presente propuesta.

A continuación, el desglose de la estructura propuesta para la fábrica:

Figura N.º 30

Estructura de la fábrica



Nota: elaboración propia.

Modelo de priorización

Las prioridades son definidas mediante el uso de un modelo de priorización que toma en cuenta variables de negocio y variables técnicas. El IMT, con base en el resultado del modelo de priorización y el análisis de capacidades, administra el tablero Kanban-Volcán (Nombre otorgado por las prácticas ágiles).

Tablero Kanban-volcán

Se propone la implementación de un tablero Kanban Volcán, el cual es una manera de priorizar las funcionalidades y ajustes. Según los resultados del diagnóstico, surge la necesidad de una solución que pudiera servir tanto a múltiples productos como a múltiples equipos.

El propósito del modelo de priorización es el siguiente:

- Priorizar el trabajo para varios equipos.
- Priorizar el trabajo para múltiples productos o familias de productos.
- Indicar la asignación de capacidad entre las diferentes familias de productos por célula.

Para las historias de usuario de proyectos nuevos o de mantenimiento, una vez se hayan construido y detallado, se define el nivel de prioridad aplicando el modelo de priorización y el IMT realiza la asignación en cada célula con la debida priorización.

Se recomienda establecer, que para los elementos del product *backlog* que correspondan a solicitudes nuevas, se debe aplicar una política de revisión de prioridad, esto es cuando el elemento se haya mantenido en el *backlog* durante dos ciclos de dos semanas, se deberá volver a aplicar el modelo de priorización y, de acuerdo con el resultado actualizado y las capacidades de las células, el IMT podrá cambiar su prioridad y ubicarlo donde corresponda.

Se proponen tres niveles de prioridad separados:

Crítico (Inmediato): este nivel es el de más alta prioridad de atención y es exclusivo para los incidentes de producción catalogados como críticos (incidentes que afectan la operatividad del servicio o el ejercicio de la funcionalidad y que atentan contra el éxito del servicio), estos son atendidos en los *fast line* de la célula de la fábrica, además, se consideran requerimientos críticos

aquellas necesidades que, si bien es cierto no son incidentes, están comprometiendo necesidades estratégicas de los clientes.

Alto (Siguiente): en este nivel se coloca el trabajo que está listo para que los equipos en las células lo tomen como lo próximo que debe hacerse.

Medio (Pronto): en este nivel se realiza el trabajo que tendrá lugar pronto. Sin embargo, para que el trabajo pase al siguiente estado, debe prepararse como se ha indicado anteriormente en el primer nivel de prioridad. (Las solicitudes catalogadas como altos y medios son categorizadas y priorizadas tanto por el *product owner* como por el IMT).

Bajo (Después): en este nivel se coloca un trabajo que puede o no ser desarrollado más tarde, en su mayoría mejoras deseables para los productos.

En cada una de las columnas definidas, cada equipo, de acuerdo con sus características y necesidades, debe tener el WIP (Work in process) limitado; esto significa no podrá haber una mayor cantidad de tareas en cada columna, que la que se defina como máximo WIP.

Debe tenerse en cuenta que la prioridad es establecer y mantener un flujo constante de trabajo terminado, así que, cuando no sea posible la colaboración entre miembros del equipo para pasar un elemento a un estado siguiente y se presenten cuellos de botella y posibilidades de tiempo ocioso de otros miembros del equipo, se debe establecer como prioridad la solución del cuello de botella y el equipo, con la ayuda del OMT, puede decidir modificar temporalmente las reglas de limitación de WIP para evitar que el sistema se detenga por completo.

Cada columna o paso del proceso debe cumplir con una definición de terminado (DoD), escrita con los criterios que deben cumplirse antes de que una tarea pueda moverse dentro de la columna. El trabajo fluye fuera del *backlog* del IMT hacia dentro de la célula, a los respectivos tableros Kanban del equipo. Cuando un equipo ha completado el trabajo (capacidad disponible), pueden ser jalados del *backlog* nuevas tareas.

A continuación, se presenta el detalle operativo necesario para la implementación de la metodología de trabajo Scrumban bajo la estructura “tipo fábrica”, incluyendo el flujo de procesos operativos, procesos de las prácticas ágiles, los roles, los artefactos (término utilizado en prácticas ágiles para las plantillas), las herramientas colaborativas y las métricas que contemplan las consideraciones relacionadas con la etapa de análisis de necesidades, desarrollo,

certificación y los eventos o sesiones propias de seguimiento del marco de trabajo que deben ser puestos en práctica dentro de los procesos.

Propuesta de procesos de desarrollo de requerimientos de *software*

La fábrica de integraciones se regirá por procesos y prácticas ágiles que garanticen la entrega de valor constante al cliente, para esto se propone la implementación de procesos que dirijan la misma hacia la mejora continua. Se identifican en la propuesta dos procesos principales para cada célula: Proceso de solicitudes nuevas y Proceso Fast Line (carril rápido).

Antes de revisar la propuesta de procesos, es necesario entender los siguientes estados:

Guía de estados

New: estado de las historias de usuario cuando ingresan a la herramienta de gestión de desarrollo

Ready: estado de las historias de usuario una vez que son aprobadas por el IMT y OMT.

Pendiente: estado de la historia de usuario reflejado como primera columna del tablero Kanban.

En proceso: estado de la historia de usuario mientras es desarrollada.

En pruebas: estado de la historia de usuario durante el proceso de certificación.

Finalizado (*Done*): estado de la historia de usuario ya aprobada por calidad.

Proceso de solicitudes nuevas

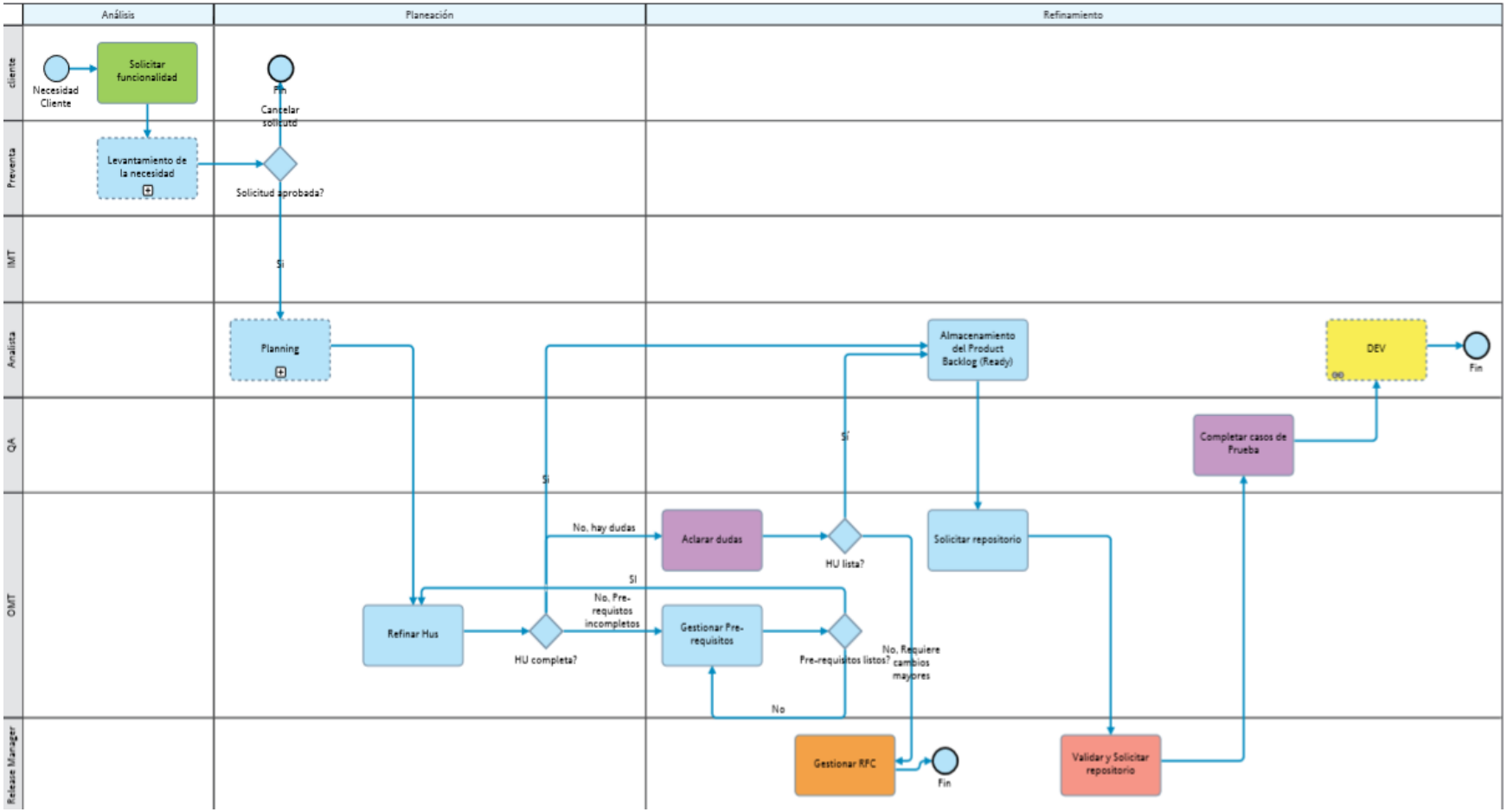
El proceso para solicitudes nuevas inicia con la solicitud del cliente, la cual es recibida por el área de preventa vía correo electrónico, quienes procederán con el levantamiento de la necesidad apoyados de recursos especializados según sea la necesidad.

Solicitud de la necesidad.

Product owner como representante del cliente solicita la necesidad, si el *product owner* no tiene claridad de lo que necesita o requiere, se le ofrece la realización de un Inception (Práctica ágil para descubrimiento de necesidades), el cual será explicado más adelante en el apartado de propuesta de metodología ágiles para la fábrica de Integration.

Figura N.º 31

Diagrama de flujo solicitudes nuevas



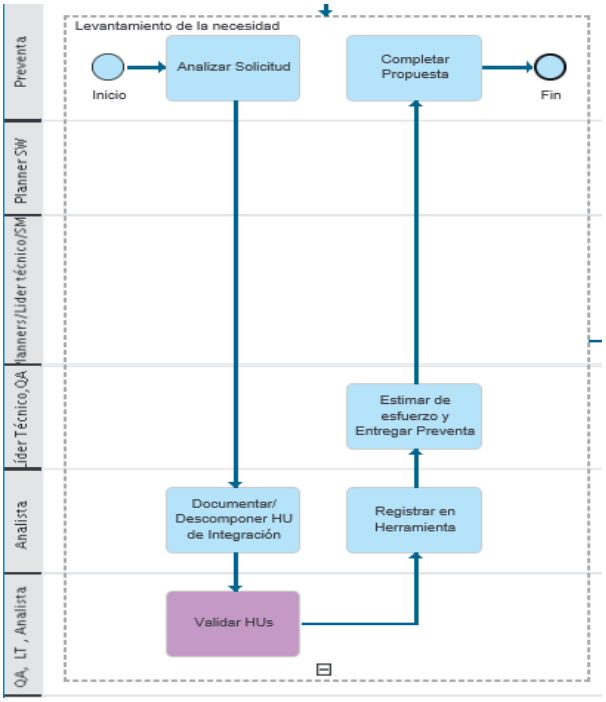
Nota: elaboración propia.

Levantamiento de la necesidad.

El equipo de preventa apoyado de un analista, un ingeniero de calidad y un líder técnico perteneciente al IMT (Integration Management Team), según se requiera, puede acompañarse de un líder técnico del OMT de la célula que maneja la línea de producto en cuestión, proceden a levantar y documentar la necesidad correspondiente mediante sesiones de trabajo (Proceso ágil Gestión de requerimientos). Una vez levantada, el ingeniero de Calidad pasa un *check list* (Apéndice No 5) para asegurar la calidad de la documentación, como equipo estiman la historia de usuario (nombre que recibe en prácticas ágiles un requerimiento); preventa afina lo necesario, se registra la o las historias de usuario en la herramienta seleccionada como repositorio como versión 1 del documento y la misma es entregada al *product owner* (Rol prácticas ágiles) para su revisión. Se entrega la historia de usuario junto con el documento de oferta respectiva (pueden ser proyectos con varias historias de usuarios o historias de usuario robustas independientes).

Figura N.º 32

Diagrama de flujo levantamiento de la necesidad



Nota: elaboración propia.

Si el proyecto o la historia de usuario es aprobada por el *product owner* (Rol de prácticas ágiles), se continúa con el proceso de *planning*, de no existir aprobación, se cierra el proceso.

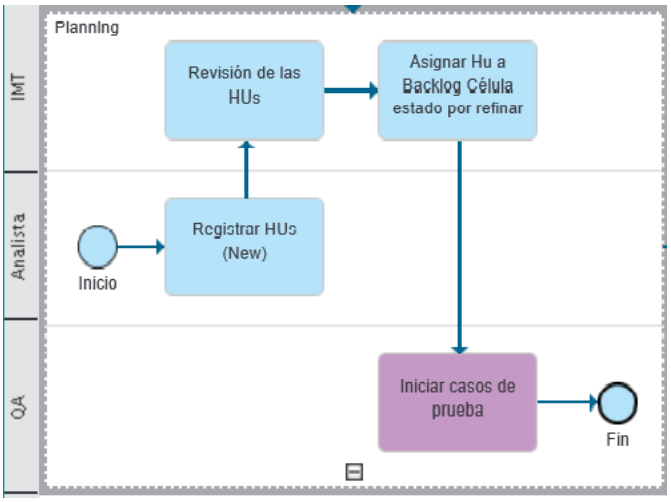
Planning.

Una vez aprobada la historia de usuario, el analista replica la o las historias de usuario desde el repositorio hasta la herramienta de gestión de desarrollos bajo el estado de *New*. El IMT revisa las historias de usuario que se encuentran en estado *New*, aplica los criterios de priorización y asigna a la célula con su debida prioridad la historia de usuario. Una vez seleccionada la célula, la historia de usuario es agregada al *backlog* de la célula correspondiente en estado *Por refinar*.

Los ingenieros de calidad de la célula pueden dar inicio con la documentación de los casos de prueba.

Figura N.º 33

Diagrama de flujo Planning



Nota: elaboración propia.

En el momento en que la historia de usuario es asignada a la célula, siempre en estado *New*, el OMT inicia su análisis mediante el proceso de refinamiento (Práctica Ágil).

Una vez finalizado el refinamiento, si la historia de usuario se encuentra completa sin pre-requisitos abiertos o dudas sin abarcar, el OMT (Operation Management Team) la coloca en estado *Ready*, lo que significa que ya la historia de usuario se reflejará al equipo automáticamente en estado *Pendiente* en el tablero Kanban. Una vez acá, ya el equipo puede disponer de la misma para su desarrollo, a su vez, cuando la historia de usuario se encuentra en *Ready*, el líder técnico del OMT procede con la solicitud del repositorio correspondiente (Práctica ágil de *release*

management). Una vez que un recurso esté disponible, se inicia el proceso de desarrollo de la historia de usuario, el o los ingenieros de calidad finalizan los casos de prueba y ejecutan los mismos una vez finalizado cada desarrollo.

Se afirma que una historia se encuentra finalizada o en estado *Done*, cuando la misma es certificada por calidad.

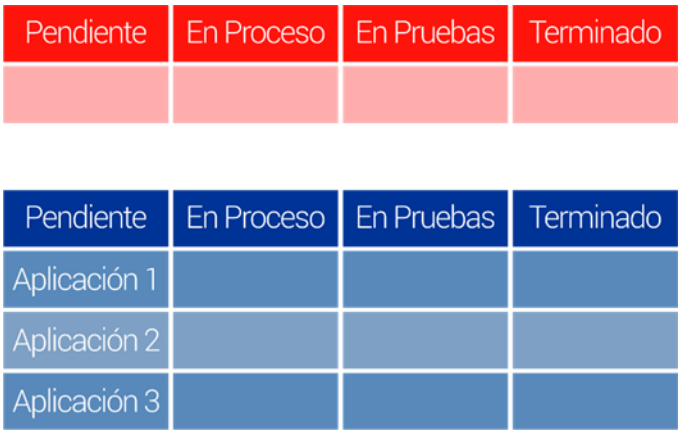
Proceso atención de tiquetes (*Fast Line*)

Se propone que el tablero Kanban de la célula cuente con un Fast Line (carril rápido) que le permita atender necesidades que surgen de último momento y el equipo tiene que atender, aunque no hayan sido planeadas. Se propone que este carril sea de uso exclusivo para atender requerimientos catalogados como críticos o estratégicos para los clientes que cuentan con bolsa de horas previamente adquiridas, se caracterizan por ser solicitudes cortas, no mayores a 8 horas de esfuerzo.

El carril rápido o *fast line* nace con el objetivo de dar continuidad y mayor rapidez a clientes que adquieren bolsas de horas y, por lo tanto, se utiliza en esquemas de mantenimiento de las soluciones previamente desarrolladas por los equipos. En el tablero Kanban convencional, se indican los pasos del proceso en forma de columnas denominadas: pendiente, en proceso, en pruebas y terminado, tanto para el Fast Line (color rojo) como para el carril de proyectos/ requerimientos nuevos.

Figura N.º 34

Tablero Kanban – Carril Solicitudes nuevas - Carril *fast line*



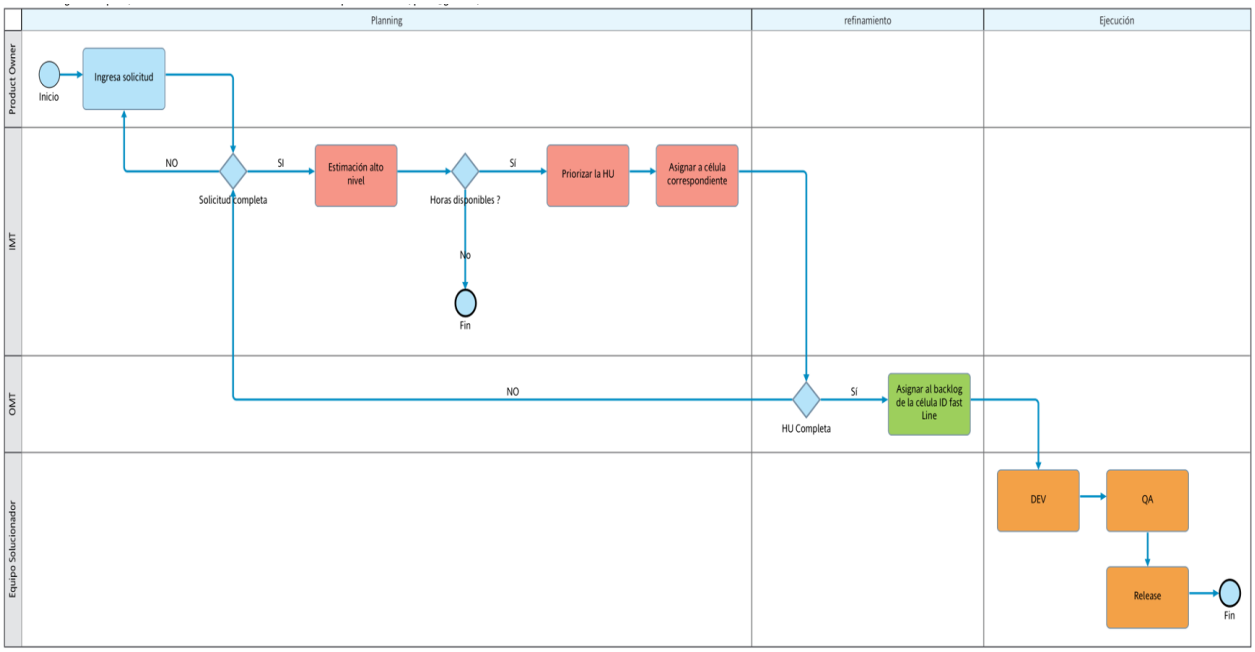
Nota: elaboración propia.

El proceso propuesto a seguir es el siguiente:

- *Product owner* ingresa la solicitud vía Service Desk (herramienta actual utilizada ya en el área) en un formato previamente establecido con el cliente (historia de usuario).
- Solicitud ingresa directamente al IMT (Integration Management Team).
- IMT verifica la completitud del contenido de la solicitud, de no estar completa, devuelve la solicitud al *product owner.*
- IMT realiza estimación a alto nivel (prácticas ágiles de estimación).
- IMT verifica la bolsa de horas disponibles del contrato previamente adquirido.
- IMT ejecuta el modelo de priorización.
- De existir horas disponibles, IMT asigna la solicitud a la célula correspondiente, de lo contrario, devuelve la solicitud y comunica al *product owner* la necesidad de un nuevo contrato.
- OMT recibe la solicitud, verifica que la historia de usuario esté completa, de no estarlo, devuelve la HU al IMT para que este vuelva a hacer la revisión y verifique la información con el *product owner;* de estar completa, la OMT la agrega al *backlog* con el identificador ID-Fast Line en la herramienta colaborativa de gestión de desarrollo.

Figura N.º 35

Diagrama de flujo carril *fast line*



Nota: elaboración propia.

Desarrollo, pruebas y entrega.

Para ambos procesos, tanto de solitudes nuevas en el carril de requerimientos como el proceso Fast Line, la propuesta define un proceso de desarrollo ágil de historias de usuario que combina el desarrollo de las historias de usuario y su proceso de pruebas en una misma iteración.

En esta etapa del proceso, se debe garantizar lo siguiente:

- Se deben seguir los estándares de desarrollo establecidos.
- Se deben desarrollar las pruebas unitarias siguiendo los estándares de cobertura de código.
- Se deben seguir los estándares de versionamiento y resolución de conflictos de código fuente.
- El desarrollador debe asegurar que se cumpla con los criterios de aceptación y los escenarios funcionales de la necesidad en el ambiente de desarrollo, antes de trasladar al ambiente de pruebas.
- El ingeniero de calidad debe realizar pruebas exploratorias en el ambiente de desarrollo.
- Se deben llevar a cabo todas las pruebas definidas en el plan de pruebas.
- Cada vez que se finalice la certificación de una historia de usuario, se presenta al *product owner* para su aprobación, validando la definición de terminado o *Done.*

Propuesta de procesos de prácticas ágiles

Para lograr la implementación de la metodología Scrumban bajo la estructura tipo fábrica, es necesario trabajar bajo prácticas ágiles, las cuales aseguren la entrega temprana de funcionalidades que agreguen valor a los clientes y a la vez, promuevan la mejora continua de los equipos de trabajo.

A continuación, se detallan las prácticas ágiles recomendadas para la presente propuesta:

Proceso ágil de gestión de necesidades

Educción y análisis- Descubrimiento de la necesidad (Inception).

Alinear y definir expectativas.

El equipo de preventa acompañado por recursos de la célula revisa con el cliente el nivel de claridad existente sobre la visión de la necesidad, de no estar claro, tal como se explicó en secciones anteriores, es necesario realizar un Inception.

Definir la visión.

Se propone incorporar un conjunto de técnicas y herramientas ágiles denominado Inception, mediante el cual y a partir de un proceso colaborativo, estructurado y aprovechando metodologías visuales, se realiza la conceptualización inicial de un proyecto; con su aplicación práctica, se obtiene una visión compartida y unificada de lo que se quiere lograr y lo que se debe construir.

Técnicas propuestas.

Tablero de visión de producto.

Obtener mediante un proceso colaborativo, una visión compartida del producto, definiendo el valor que se va a entregar a cada grupo de usuarios atendiendo sus necesidades mediante las funcionalidades de alto nivel.

Delimitar y priorizar.

Se delimita de forma clara y priorizada su alcance, en términos de necesidades o historias de usuario, entre otros elementos que configuran un *product backlog* (Lista de necesidades del producto).

El propósito del Inception es precisamente definir ese norte que guía al proyecto, llevando a cabo todas las actividades previas al inicio del desarrollo de las necesidades. Así, se busca que el tiempo sea lo más corto posible, desde que se concibe inicialmente la idea del proyecto o de la iniciativa hasta que se inician los desarrollos, tomando en cuenta los riesgos a partir de etapas tempranas.

Técnicas propuestas.

El vecindario.

Definir los interesados relevantes y cómo son impactados por el producto o cómo pueden impactarlo.

La lista del NO.

Definir qué NO es el producto, entender que puede haber expectativas y funcionalidades que podrían identificarse y que no han sido contempladas en el alcance del proyecto hasta ahora. El aporte fundamental de esta técnica consiste en generar conversaciones y conclusiones con respecto al negocio y a la generación de valor. Ayuda a mantener visible y claro el horizonte del proyecto y de la solución para todos.

Mapa de historias de usuario.

Definir las funcionalidades de alto nivel mediante la construcción del mapa de historias de usuario. Como resultado de este ejercicio, se obtiene un mapa de historias de usuario completo, que contiene tanto las historias de usuario, como los requerimientos técnicos, sobre el cual se puede sacar la información para planear con mayor nivel de detalle el tiempo de ejecución del proyecto, los recursos, el número de personas necesarias, las capacidades de las personas y las herramientas tecnológicas para implementar la solución.

Mínimo producto viable.

Priorizar las funcionalidades y definir el mínimo producto viable, se define el mínimo producto viable a partir del entendimiento de las funcionalidades a alto nivel y la priorización en función del valor a entregar. Es lo mínimo que el *product owner* necesita para poder poner a disposición de sus usuarios el producto.

Plan de Salidas a producción (releases).

Agrupar las funcionalidades que no están en el mínimo producto viable en *releases* candidatos, se define a partir de la generación de valor incremental, el plan de *releases* o entregas de versiones de la aplicación.

Elaborar historias de usuario

Partiendo de una lista de historias de usuario, construidas en el mapeo de historias de usuario y asociadas a la visión del producto definido en Inception, se procede con la construcción de historias de usuario detalladas, para lo cual el IMT (analista de negocio, el ingeniero de calidad, el arquitecto de solución y el *product owner*) se reúnen antes de iniciar la primera iteración, con el fin de construir las historias más prioritarias y, durante el proyecto, ir detallando las historias en la medida que su prioridad va aumentando.

Los pasos para elaborar una historia de usuario son los siguientes:

Entendimiento de la necesidad: el analista se encarga de entender el problema, el negocio y las necesidades del solicitante, está información puede ser suministrada por el *product owner*.

Construcción base de la necesidad: el analista, documenta junto con el *product owner* las necesidades en forma de historias de usuario y los criterios de aceptación en el formato llamado historia de usuario.

Detalle de la historia de usuario: el analista, el *product owner*, el líder técnico, el ingeniero de calidad y el arquitecto de solución entienden las historias de usuario y definen los aspectos técnicos, de diseño de arquitectura y de calidad para el cumplimiento de los criterios de aceptación. El equipo debe establecer una estimación previa del esfuerzo requerido para cada elemento. En caso de ser necesario, además de las historias de usuario, se escriben los requerimientos técnicos identificados

Validación técnica y creación o ajuste del diseño de arquitectura: el líder técnico y el arquitecto de solución validan las consideraciones técnicas y diseño de arquitectura que conlleva cada historia de usuario para su construcción. Cada uno documenta en el formato correspondiente los detalles necesarios.

Construcción de la matriz de dependencias: una vez analizados los aspectos técnicos, de arquitectura y de calidad, se definen las dependencias con otras aplicaciones o funcionalidades

que conlleva el desarrollo de la solución. El analista debe indicar las dependencias funcionales de las historias de usuario y el arquitecto de solución con el líder técnico deben indicar las dependencias técnicas de las historias de usuario.

Especificar los criterios de aceptación.

Cada historia de usuario debe incluir la documentación de los criterios de aceptación, basado en estos criterios, el equipo desarrollará cada solución y constituyen la guía para que los ingenieros de calidad puedan revisar lo desarrollado.

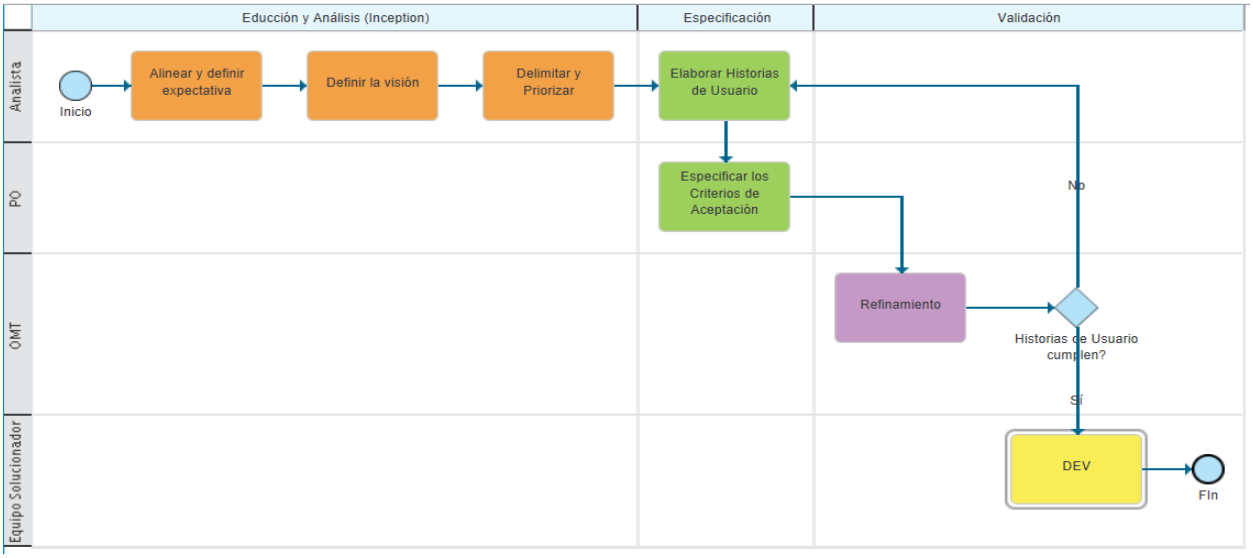
Refinamiento

El refinamiento es un evento o sesión presente a lo largo de todo el ciclo de desarrollo, se realiza con el objetivo de revisar las historias de usuario, aclarar dudas, garantizar su completitud y llevarla a estado de Ready para que el equipo la pueda trabajar, con una duración no mayor a dos horas(Tiempo recomendado según la guía de Scrum y las buenas prácticas de la industria), y al menos una vez por semana este evento garantiza la continuidad de operaciones de la célula.

Participa el OMT, incluyendo al *product owner,* quien debe presentar el product *backlog* ya priorizado según el valor que le genera cada historia de usuario en cada uno de los eventos de refinamiento. Si en el refinamiento se define que la historia de usuario está completa y lista para ser desarrollada, la misma se coloca en *Ready*.

Figura N.º 36

Diagrama de flujo Gestión de requerimientos



Nota: elaboración propia.

Control y seguimiento

Para el área en estudio, se decide mantener los eventos o sesiones de seguimiento y mejora continua que propone el marco Scrum, de la siguiente manera mostrada en la figura 37.

Figura N.º 37

Eventos o ceremonias para el seguimiento

Nota: elaboración propia.

Daily Scrum Meeting.

El equipo Scrumban (equipo solucionador y Scrum máster) se reúne a entablar una conversación corta y concisa sobre el avance del día anterior. En caso de que haya impedimentos, ya sea para la actividad que se está ejecutando o que se va a empezar a ejecutar, se debe comentar en el equipo para identificar alternativas de soluciones grupales, con el fin de sobrepasar esta situación. Si la solución es externa y no se encuentra dentro de las personas del equipo Scrum (equipo solucionador, Scrum máster y *product owner*), el Scrum máster toma ese impedimento como una actividad propia para facilitar su resolución lo más pronto posible. Cuando no se pueda resolver, se debe tener claro qué impacto puede tener este para la entrega del *sprint* o requerimientos urgentes- incidentes y de esta forma gestionar alternativas con el equipo.

Los objetivos de la reunión y las preguntas sugeridas son las siguientes:

Objetivo general del evento: incrementar la comunicación, explicitar los compromisos, dar visibilidad a los impedimentos.

Pregunta sugerida: ¿Qué he hecho desde la última reunión y si pude hacer todo lo que tenía planeado?

Objetivo de la primera pregunta: actualización.

Pregunta sugerida: ¿Qué voy a hacer a partir de este momento?

Objetivo de la segunda pregunta: sincronización.

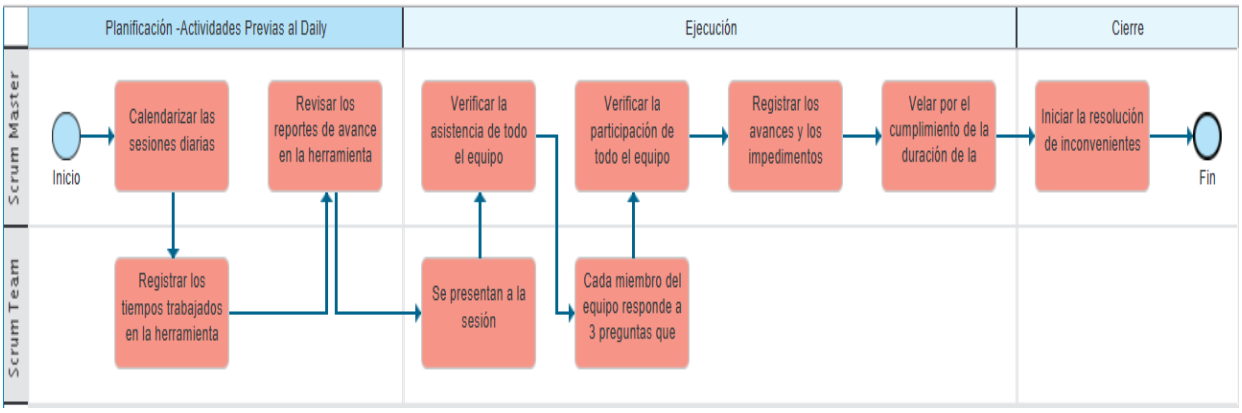
Pregunta sugerida: ¿Qué impedimentos tengo o voy a tener para cumplir mis compromisos en esta iteración y en el proyecto?

Objetivo de la segunda pregunta: detección temprana de impedimentos.

Duración aproximada del evento: 15 minutos, tiempo estipulado en la Guía de Scrum (Schwaber y Sutherland, 2017). Se propone la realización de la misma a las 9:00 a.m.

Figura N.º 38

Diagrama Daily Scrum Meeting



Nota: elaboración propia.

Sprint Review (Revisión de la iteración).

El Scrum máster cita al evento y facilita la sesión, en la que se presentan y analizan los resultados de la iteración, basándose principalmente en los indicadores definidos para el equipo. Además, se hace un resumen de los elementos del *product backlog* que fueron desarrollados y finalizados durante la iteración, revisando también cómo fue distribuida la capacidad del equipo en los diferentes tipos de elementos.

Esta reunión, en la que participa todo el equipo solucionador y el OMT, donde el actor principal es el *product owner*, se lleva a cabo cada dos semanas, preferiblemente en el mismo lugar y a la misma hora, para generar la disciplina necesaria, las entradas son:

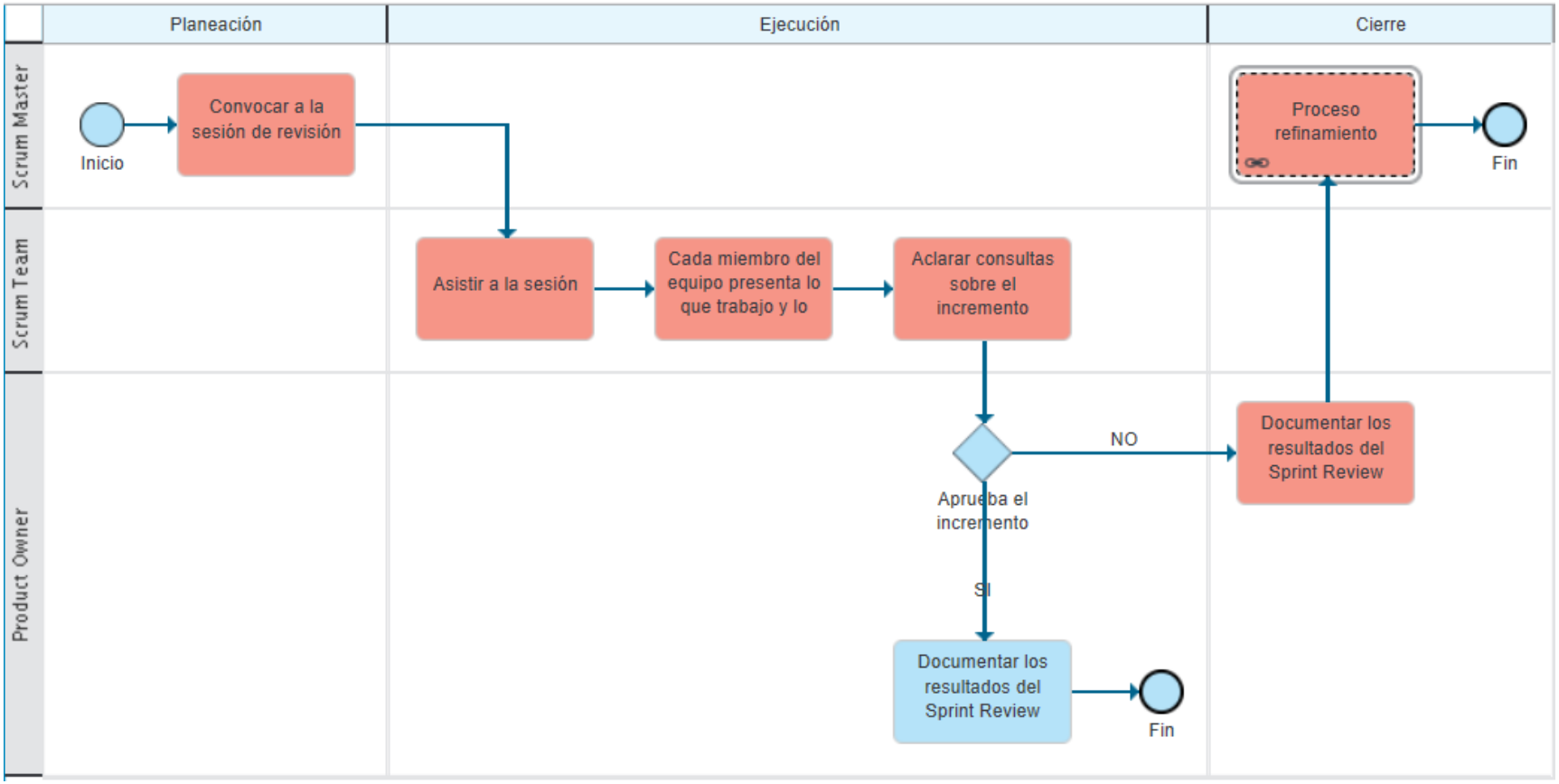
- Resumen de los indicadores del equipo.
- Resumen de las necesidades terminadas durante la iteración.
- Información relacionada con la asignación real de capacidad del equipo para los diferentes tipos de necesidades.

Las salidas son:

- Retroalimentación y acciones de mejora relacionadas con los indicadores.
- Posibles acciones de mejora relacionadas con la capacidad para atender diferentes tipos de necesidades.

Figura N.º 39

Diagrama de flujo *Sprint review*



Nota: elaboración propia.

Sprint retrospective.

Todas las retrospectivas se realizan para evaluar el proceso, el producto o las personas. Aunque hay muchas maneras de llevar a cabo un evento de retrospectiva, la manera más simple es la siguiente:

- Evaluar los aspectos positivos.
- Trazar planes de acción para mantener y seguir potenciando estos aspectos.
- Determinar los aspectos por mejorar.
- Establecer planes de acción para mejorar.
- Asignar las tareas en el equipo como compromiso para velar por la mejora de estos aspectos.

Otro tipo de retrospectiva corta recomendada es que cada integrante responda:

- ¿Qué empezar a hacer?
- ¿Qué debe dejar de hacer?
- ¿Qué debe seguir haciendo?

Hay muchas variaciones en este formato simple. El Scrum másster puede facilitar esta reunión retrospectiva de la iteración pidiendo a los participantes que solo den ideas. El Scrum máster puede ir alrededor de la habitación pidiendo a cada persona que identifique temas en común que se deban iniciar, detener o continuar.

Después de una primera lista de ideas, los equipos votarán sobre temas específicos para tratar durante la próxima iteración. Al final de la iteración, la próxima retrospectiva a menudo comienza revisando la lista de cosas seleccionadas para la anterior retrospectiva de la iteración.

La retrospectiva será la última actividad que se realice en una iteración definida. Muchos equipos lo harán inmediatamente después del Sprint review. Todo el equipo debe participar. Se puede programar una retrospectiva de Scrum que dure hasta una hora, lo cual es suficiente según la guía de Scrum (Schwaber & Sutherland, 2017). Sin embargo, de vez en cuando un tema candente se presentará o un conflicto de equipo se intensificará y la retrospectiva podría tomar mucho más tiempo.

Entradas para el evento de retrospectiva:

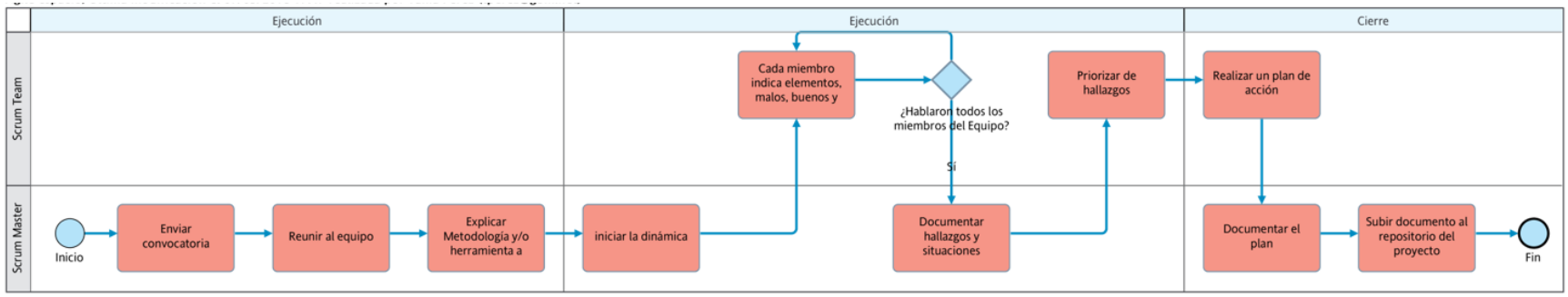
- Aspectos positivos identificados por cada integrante durante la iteración.
- Aspectos negativos identificados por cada integrante durante la iteración.

Salidas para el evento de retrospectiva:

- Plan de acciones de mantenimiento y potenciación de aspectos positivos.
- Plan de acciones de mejora.
- Asignación de líderes del equipo para velar por el mantenimiento de las buenas prácticas y las acciones de mejora.

Figura N.º 40

Diagrama de flujo Retrospectiva



Nota: elaboración propia.

Proceso ágil de gestión de la calidad

Se propone como un aspecto esencial, la participación e involucramiento del área de Calidad del Software desde las etapas tempranas de los desarrollos y durante toda la ejecución de los mismos. Por este motivo, se propone un esquema ágil de pruebas de *software*, el cual contempla todas las labores de verificación y validación realizadas al producto de *software* construido o ajustado, de tal forma que se pueda asegurar que los resultados satisfacen las necesidades de la organización en términos de calidad de producto entregado.

Es necesario entonces que el área de Calidad de Software participe en las etapas previas al inicio del desarrollo de las necesidades, para facilitar el entendimiento de estas y la correcta administración de las tareas de pruebas necesarias, de esta forma, durante el Inception, el ingeniero de calidad debe encargarse de asegurar la revisión y definición a alto nivel de los siguientes puntos, así como la inclusión de los mismos en el informe de la o las sesiones de Inception:

-Elementos de pruebas: se deben identificar a alto nivel todas aquellas piezas de *software* que serán probadas en el proceso de control de calidad; puede incluir soluciones de *software* completas, módulos, o servicios.

-Nuevas funcionalidades a probar: se deben identificar todas las funcionalidades que serán cubiertas en el proceso de control de calidad.

- Pruebas de regresión: se debe identificar la necesidad de realizar pruebas de regresión para asegurar que no existan daños colaterales dada una reparación o corrección de *software*.

-Funcionalidades a No Probar: se deben identificar todas las funcionalidades que se excluirán del proceso de control de calidad.

-Estrategia de pruebas: se debe definir una visión a alto nivel del enfoque que seguirá el proceso de control de calidad; incluyendo el alcance, objetivos, enfoque de pruebas, entregables del proceso de control de calidad, enfoque del seguimiento de defectos, y la identificación de entrenamiento, automatización y riesgos.

-Requerimientos de Entornos – *Hardware*: se debe identificar el listado de todas las necesidades de *hardware* requeridas para llevar a cabo el proceso de control de calidad, como, por ejemplo: servidores, balanceadores, entre otros.

-Requerimientos de Entornos – *Software*: se debe definir el listado de todas las necesidades de *software* requeridas para llevar a cabo el proceso de control de calidad, como, por ejemplo: licencias, sistemas operativos, integraciones con *software* de terceros, entre otros.

- Herramientas de pruebas requeridas: se debe definir un listado de herramientas necesarias para soportar el proceso de control de calidad, por ejemplo: gestión de las pruebas, herramientas de pruebas no funcionales para rendimiento o seguridad.

-Colaboradores: se debe realizar la identificación de la cantidad de personal, y roles requeridos, para llevar a cabo el proceso de control de calidad.

-Entrenamiento: se refiere a la identificación de las necesidades de entrenamiento que necesite el personal del área de Calidad de *Software* para llevar a cabo sus labores en el proyecto; el entrenamiento puede estar enfocado a la solución de *software* a probar o a herramientas especializadas requeridas durante el proceso de pruebas.

Se enumera a continuación una lista de características de pruebas según los tipos de necesidades o proyectos y el marco de trabajo utilizado en cada uno de ellos.

Tabla N.º 20

Características de pruebas

Criterio	Proyectos Solicitudes nuevas y Fast Line (Scrumban)	Soporte
Plan de pruebas en herramienta de trabajo	Sí	Opcional
Estimación	Sí	Opcional
Diseño de pruebas	Basado en riesgo asociado al cambio	Basado en impacto en producción

Nota: elaboración propia.

Adicionalmente, se definen una serie de criterios para los casos de prueba que determinarán la aceptación, la suspensión o la reanudación de los mismos:

Criterios de aceptación: determinan, a través de su validación, que un caso de prueba puede ser aceptado o rechazado.

Criterios de suspensión: determinan, a través de su validación, que un caso de prueba no puede ser ejecutado y debe ser suspendido, por ejemplo, por la falta de datos de pruebas u otros factores varios.

Criterios de reanudación: determinan, a través de su validación, que un caso de prueba que había sido suspendido previamente puede ser reanudado para su correcta ejecución.

Además, en los procesos establecidos se realiza la definición de los entregables necesarios a lo largo de todo el ciclo de vida del desarrollo de solución de necesidades para el negocio, un entregable se define como toda aquella documentación que se genere producto del proceso de pruebas, se han definido al menos los siguientes elementos: estrategia de pruebas, casos de pruebas, evidencias de ejecución y visto bueno de pruebas.

Se propone que en todo proyecto imperará el criterio del ingeniero especialista en control de la calidad del *software*, para la elección del tipo de pruebas por realizar, de forma tal, que se asegure que la calidad definida e identificada para cada proyecto sea la más adecuada según su naturaleza.

Los tipos de pruebas a incluirse pueden abarcar las siguientes categorías:

- Pruebas funcionales: son aquellas orientadas a asegurar el cumplimiento a cabalidad de las especificaciones funcionales determinadas por el cliente o dueño del producto, asegurando así su aceptación y satisfacción. Podría incluir los siguientes tipos:
    - Pruebas exploratorias: prueba con poco planeamiento, y con enfoque de exploración sobre la solución de *software* a probar.
    - Pruebas de *smoke*: pruebas enfocadas a asegurar la idoneidad de la funcionalidad crítica.
    - Pruebas de *build*: pruebas que incluyen un conjunto de pruebas regresión y además conllevan pruebas para asegurar la idoneidad de la nueva funcionalidad liberada.
    - Pruebas de regresión
- Pruebas no funcionales: son aquellas orientadas a asegurar el cumplimiento a cabalidad de las necesidades especiales determinadas por el cliente o dueño del producto, asegurando así su aceptación y satisfacción. Se listan algunos tipos, más podrían aplicarse otros dada la necesidad específica de la solución de *software*:

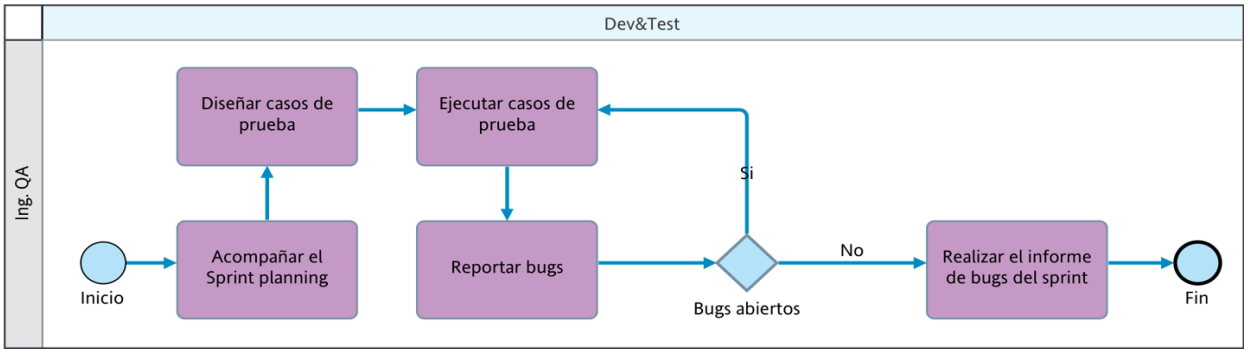
- Pruebas de usabilidad: pruebas enfocadas en verificar la facilidad de uso de una solución de *software*.
- Pruebas de rendimiento: pruebas enfocadas en verificar que los tiempos de respuesta sean aceptables.
- Pruebas de seguridad: pruebas enfocadas en verificar el comportamiento de una solución de *software* ante un ataque malicioso.
- Pruebas de recuperación: pruebas enfocadas en verificar que tan rápido puede recuperarse una solución de *software* luego de un fallo del sistema o de *hardware*.
- Pruebas de conformidad: pruebas enfocadas en validar el apego de una solución de *software* a un estándar definido.

Propiamente durante el *sprint* o iteración definida, los recursos con habilidades de gestión de la calidad participan durante el evento de planeación, diseñan casos de prueba, una vez iniciado el desarrollo, el equipo de calidad inicia la ejecución de las pruebas, reportan defectos y documentan un informe de defectos para la iteración.

Se detalla a continuación el flujo de actividades:

Figura N.º 41

Diagrama de gestión de la calidad



Nota: elaboración propia.

Proceso ágil de *Release management*

Se presenta la propuesta de procesos de administración de versionamiento y ramificación para la fábrica de *software* de Integration, enfocándose en la definición de un marco de trabajo general para las actividades de la fábrica a nivel de conceptos, estrategia de ramificación, procesos detallados de la metodología, métricas, así como roles (ver apartado propuesta de roles de la presente propuesta) y responsabilidades, todo lo anterior alineado al proceso definido para desarrollo con la metodología Scrumban en la fábrica, con el objetivo de gestionar la entrega y el proceso de versionamiento de todo *software* nuevo o de actualizaciones de *software* tanto en fases de desarrollo como de producción.

Arquitecturas de almacenamiento.

Existen dos formas de realizar el control de versiones:

Distribuidos: cada usuario tiene un repositorio local, los distintos repositorios pueden intercambiar y mezclar revisiones entre ellos. Es frecuente el uso de un repositorio, que está normalmente disponible y sirve como punto de sincronización de los distintos repositorios locales.

Centralizados: en este tipo de control de versiones existe un repositorio centralizado de todo el código, del cual es responsable un único usuario (o conjunto de ellos). Se facilitan las tareas administrativas a cambio de reducir flexibilidad, pues todas las decisiones fuertes (como crear una nueva rama) necesitan la aprobación del responsable.

Para la fábrica de Integraciones, se propone la utilización de un control de versiones centralizado.

Gestión de versiones.

La gestión de versiones propuesta se encargará de la implementación y control de la calidad del *software* instalado en el entorno de producción. La misma debe poner a prueba e instalar las versiones en el entorno de producción con los cambios preestablecidos.

Clasificación de versiones.

Se propone la siguiente clasificación de versiones:

Versiones mayores: representan importantes modificaciones en la funcionalidad del *software*, características técnicas, entre otros.

Versiones menores: representan corrección de errores puntales.

Versión emergencias: representan correcciones en producción.

El sistema universalmente aceptado es:

Versiones mayores: 1.0, 2.0, etc.

Versiones menores: 1.1, 1.2, 2.1, 2.2, etc.

Versiones de emergencias: 1.1.1, 1.2.1,1.2.2, etc.

Elementos básicos de versionamiento.

Como se explicó en el apartado anterior, el sistema de control de versiones centralizado es en el que se usa un repositorio único y compartido para mantener una copia del código fuente y todos los cambios se hacen contra esa copia central.

Los desarrolladores deben tener copias locales de los archivos que modifican y deben enviar de nuevo al servidor dichos cambios, para compartirlos con los demás miembros de su equipo. A continuación, se muestran elementos básicos de versionamiento que el desarrollador debe tener en cuenta a la hora de subir el código al repositorio de GBM.

Línea base.

Se debe realizar una revisión aprobada del código fuente, a partir de la cual se pueden realizar cambios subsiguientes.

Branch (Rama).

Se debe crear un *branch* antes de empezar a modificar el código, los *branch* son ramas que tienen una ubicación diferente al código fuente inicial, estas ramas contienen partes de código (archivos) que se encuentra en desarrollo, el cual permite a dos o más usuarios trabajar en los diferentes *branch.*

Merge.

Se debe ejecutar el proceso de *merge*, donde se toma el código de los diferentes *branch* y se realiza una unificación. Para realizar un *merge*, se selecciona la fuente origen y la fuente destino.

Checkout.

Al inicio del proyecto, se debe bajar el proyecto desde el repositorio a un directorio local. Esta operación se realiza solo una vez.

Commit.

Todos los desarrolladores deberán hacer *commit* según la frecuencia definida entre el *release manager* y el equipo, para la presente propuesta, se establece la realización de un *commit* diario por desarrollador. Es decir, cada desarrollador deberá realizar al menos una confirmación de cambios que se están realizado al día, donde se solicita agregar un mensaje con una descripción de los cambios realizados.

Subir cambio (Check in, push).

Una vez que se realizó el *commit* en el *branch*, se debe actualizar la versión o los archivos que se encuentran en el servidor, esto se realiza por medio de una validación de los archivos que están pendientes de subir.

Update (Fetch).

Se deben traer los cambios realizados del repositorio remoto al directorio local, esto se hace una vez que se haya realizado el *merge*.

Historial de cambios.

Se debe manejar un historial de cambios, donde se visualizan los cambios que se han realizado en el proyecto, muestra quién realizó el cambio y en qué fecha. Esto ayuda al equipo a validar todos los cambios del proyecto durante el proceso.

Conflictos.

Ocurren cuando existen muchos cambios en el código en el mismo sitio y el repositorio reconoce las diferencias, pero no sabe cómo resolverla.

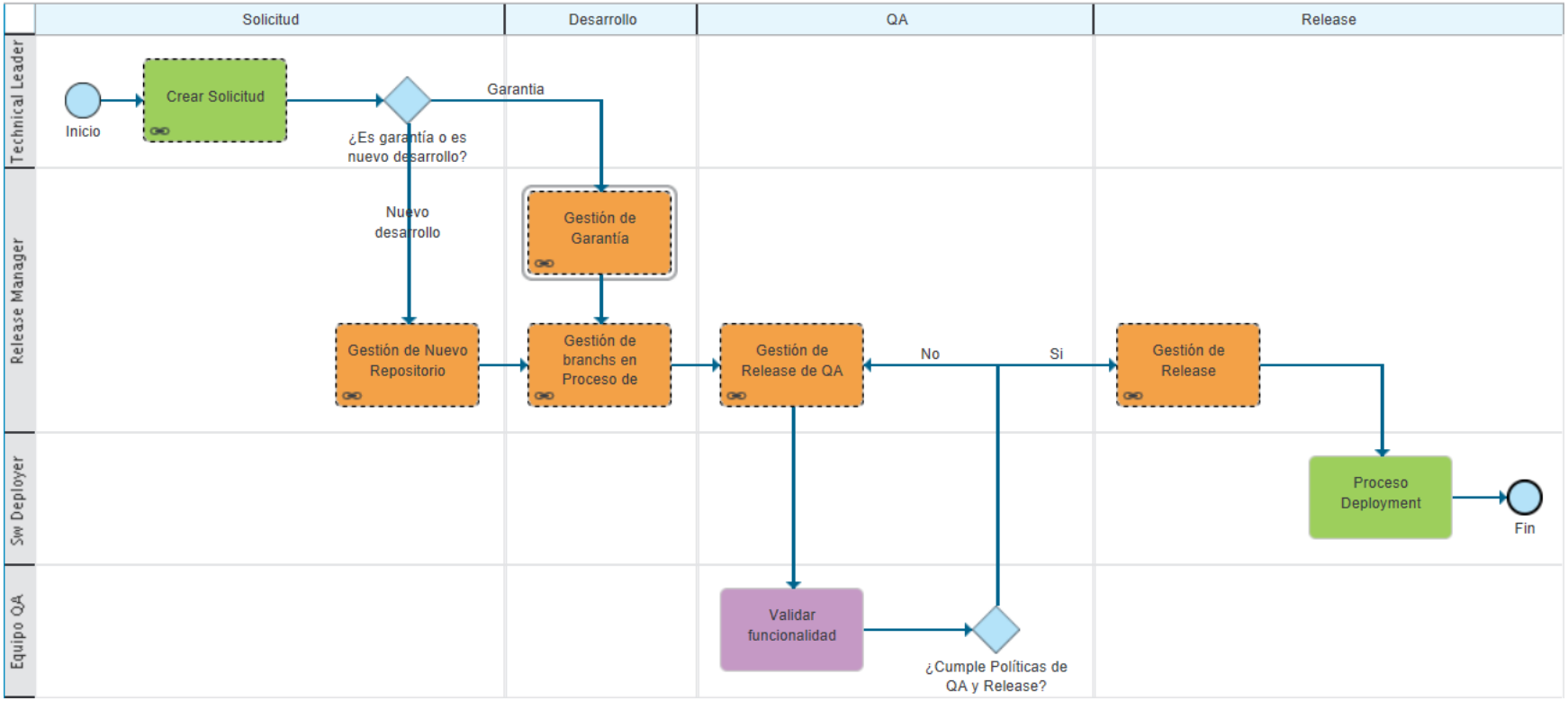
El sistema le muestra al usuario las siguientes opciones:

- Seleccionar la opción de origen.
- Seleccionar la opción de destino.
- Descartar ambas opciones
- Realizar una combinación de los cambios para continuar el proceso de merge.

A continuación, se presentan los diagramas de flujo que muestran los procesos antes mencionados.

Figura N.º 42

Diagrama de flujo *Release management*



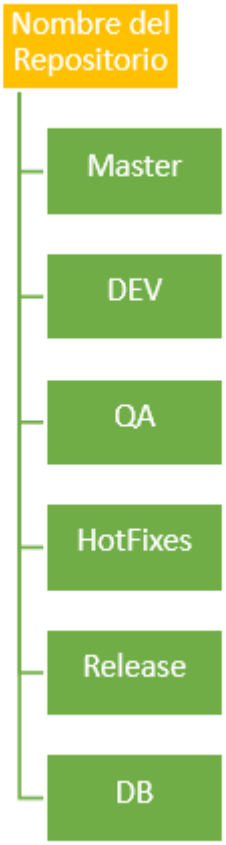
Nota: elaboración propia.

Estructura de versionamiento.

A continuación, se muestra la propuesta de estructura de *branch* para el proceso de *release management* de la fábrica (Figura 43).

Figura N.º 43

Estructura de versionamiento Integration



Nota: elaboración propia.

1. *Branch* de máster: en esta rama se encuentra la línea principal del proyecto.
2. *Branch* de desarrollo (DEV): en esta rama se van crear todos los *branch* que se requieren para ir realizando los nuevos desarrollos, mejoras o agregar funcionalidades para el siguiente *release.*

Figura N.º 44

Ejemplo *branch* de desarrollo

Nota: elaboración propia.

Si se está trabajando en un desarrollo con múltiples funcionalidades a la vez, cada una de estas debe tener un *branch* separado, esto con el fin de que los cambios que realizan los desarrolladores no sean dañados por otro que está trabajando en algo distinto.

3. *Branch* de QA: en esta rama se van a encontrar las funcionalidades que ya pasaron el proceso de QA y que requiere validar que cumplen con las políticas de RM para realizar el *release* final.
4. *Branch* de *hot fixes*: aquí se van a crear los *branch* pasa solucionar problemas de emergencia que se realicen cuando la aplicación esté en producción.
5. *Branch de release*: en esta rama se encuentran las funcionalidades terminadas y testeadas.
6. *Branch* de DB: aquí se van a agregar los *scripts* de base de datos.

Nomenclatura de los Branch.

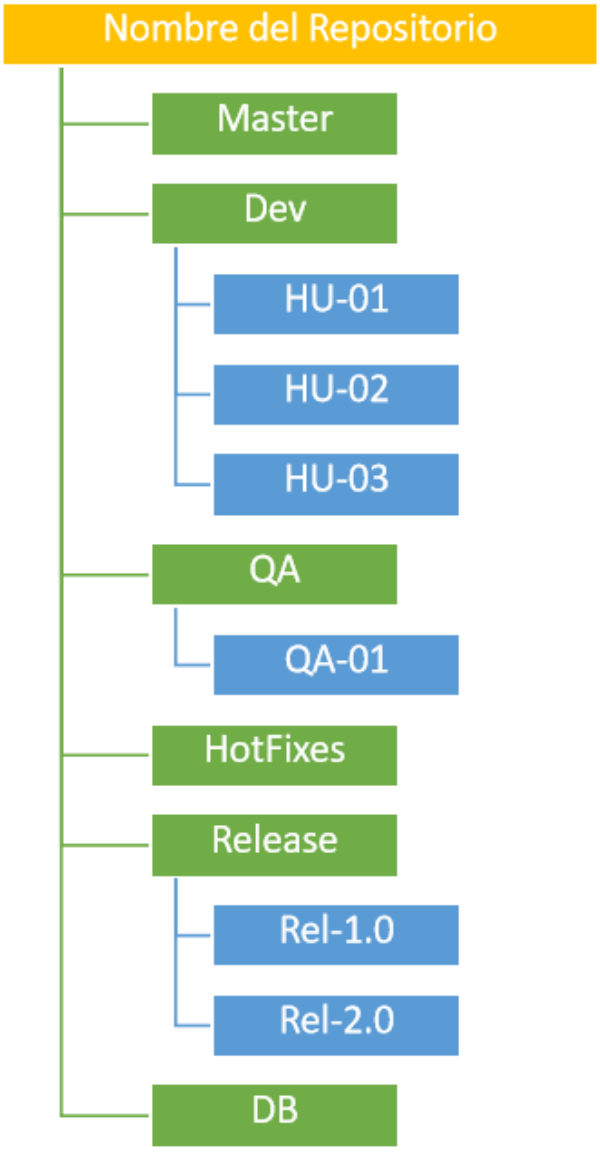
• Para los *branch* que se van a crear en el *branch* de Desarrollo (DEV), se define que este va llevar el prefijo y el número consecutivo de la historia de usuario, por ejemplo: HU-01, HU-02, QA-01.

• Cuando se realice un *release*, se debe aplicar una nomenclatura diferente, debe de llevar un prefijo “Rel” y un número consecutivo (ver el punto de Clasificación de versiones).

Esquema de ejemplo:

Figura N.º 45

Ejemplo uso de nomenclatura en los *branch*



Nota: elaboración propia.

Políticas de *Release management.*

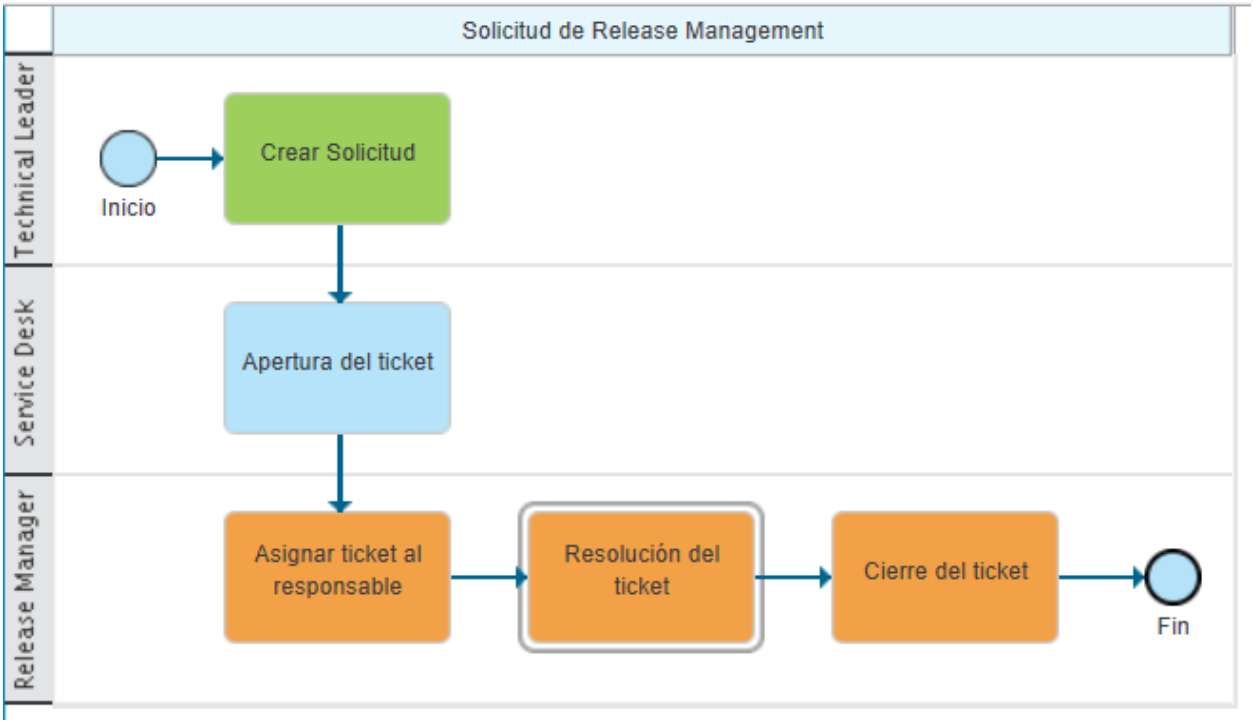
Se definen como las reglas para establecer un buen control y seguimiento de las versiones que se van a realizar en los proyectos las siguientes:

Manejo de carpetas.

1. Cuando se inicie con un nuevo desarrollo, el líder técnico de la OMT debe enviar la solicitud de crear un nuevo repositorio al *release manager*, donde se albergará el nuevo proyecto.

Figura N.º 46

Subproceso solicitud de *Release management*



Nota: elaboración propia.

2-Una vez que esté creado el repositorio, los equipos de desarrollo deberán trabajar en el *branch* principal de Dev, dentro de este, se deberán crear *sub branch*, donde cada integrante del equipo tendrá su *branch* para trabajar en la nueva funcionalidad o cambio (Ver Políticas de creación y manejo de *branch*).

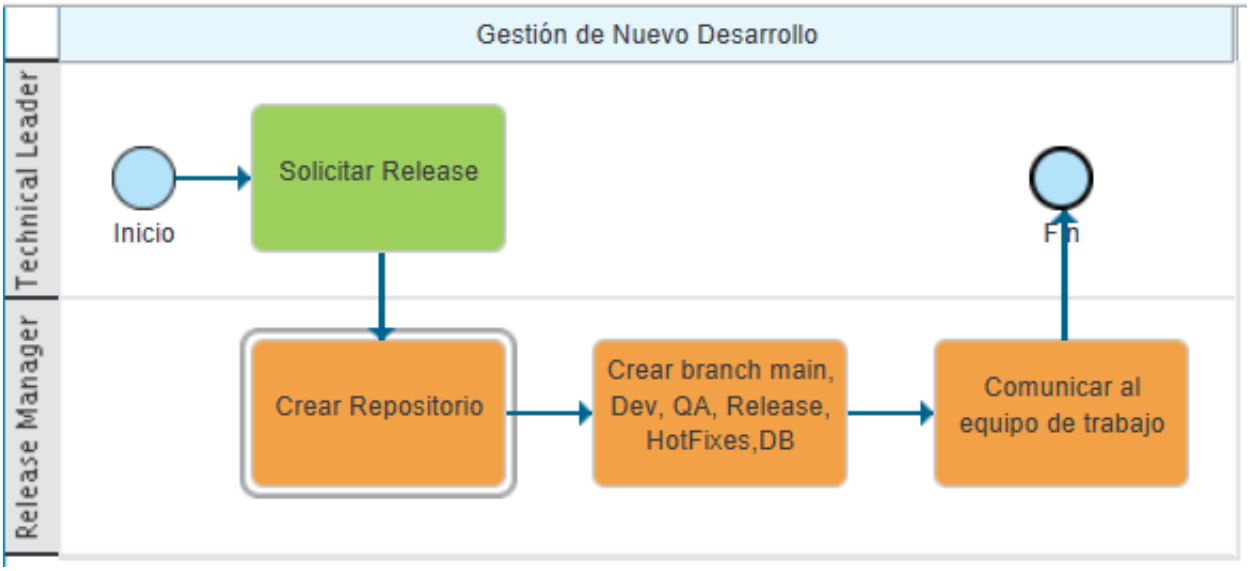
3. Una vez finalizado el proceso de desarrollo, el líder técnico de la OMT debe enviar una solicitud de crear *release* de QA al *release manager* y este *release* será entregado al ingeniero de QA para las pruebas requeridas.

4. Si el desarrollo cumple con las pruebas, el ingeniero de QA debe enviar la aprobación al *release manager*, para que este realice el *release* correspondiente, ya sea a un ambiente en producción o al ambiente de pruebas del *product owner*.

5. Una vez que un *release* ya esté implementado y se presente algún incidente que se requiera atender de inmediato, se deberá hacer un *branch* en *hot fixes* y ahí hacer un *merge* con lo que está en *release* y hacer el cambio correspondiente. El equipo del proyecto luego deberá comunicar el cambio que se realizó al *release manager* para que genere una nueva versión.

Figura N.º 47

Diagrama de flujo Gestión de Carpetas para Nuevos Desarrollos



Nota: elaboración propia.

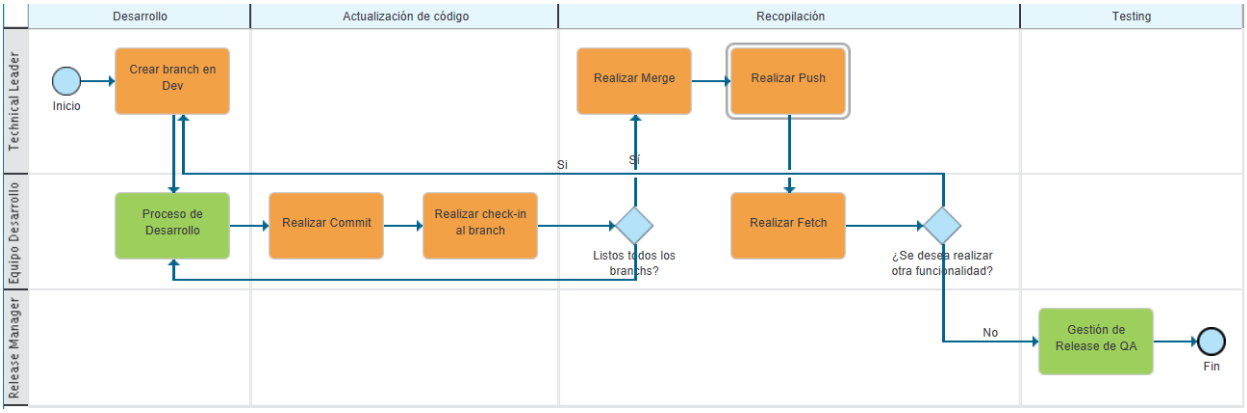
Políticas de creación y manejo de branch.

1. Para crear los sub *branch* en la carpeta de Dev, el líder técnico debe solicitar al *release manager* la creación de los mismos.
2. El líder técnico debe crear en el repositorio los *branch* que requiera el equipo de desarrollo. La creación de los *branch* se va a realizar siempre que haya una nueva funcionalidad o para corrección de errores.

3. Cuando se realice un nuevo *branch*, este debe cumplir con la nomenclatura establecida en la metodología de *release management.*
4. Los equipos deben realizar un *commit* y *check-in* cada que vez que finalicen una funcionalidad, en el caso de que la funcionalidad no esté lista, se debe realizar al final de la jornada laboral (4:00 p.m.), esto con el fin de que quede en el *branch* del repositorio una copia del trabajo realizado, aunque este no esté completo o contenga *bugs*.
5. Cuando se realice un *commit*, el comentario o descripción que se agregue debe estar asociado al cambio o la funcionalidad desarrollada. Los *commits* descriptivos hacen la revisión del código un trabajo mucho más fácil.
6. Validar que los *branches* creados siempre estén actualizados.
7. El encargado de realizar los *merge* en el *branch* de Dev, es el líder técnico de la OMT.
8. Para hacer un *merge* de las *branch*, las funcionalidades deben tener realizadas las pruebas unitarias.
9. Realizar *Update(Push)* en los repositorios locales cada vez que se actualice el repositorio remoto.

Figura N.º 48

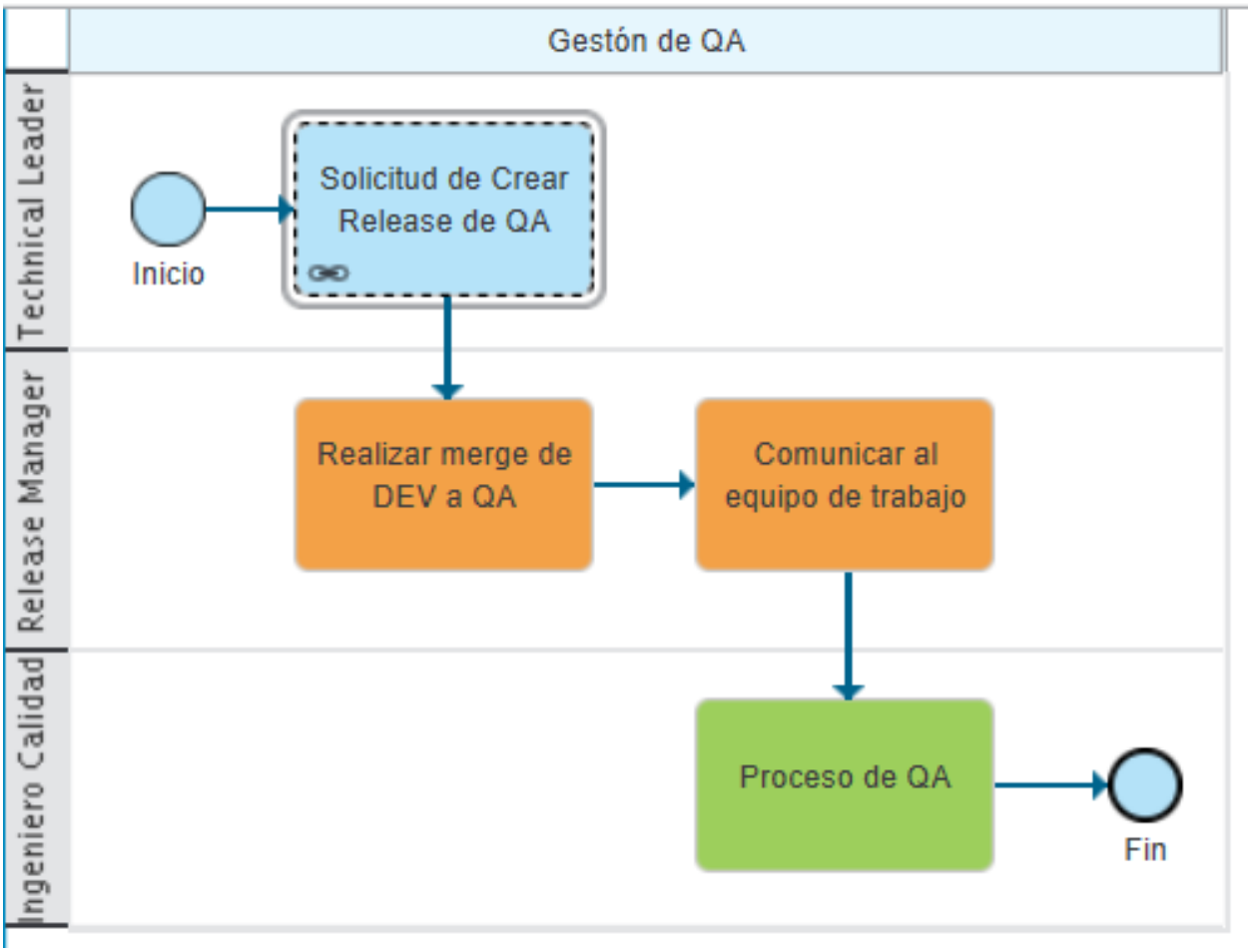
Diagrama de flujo manejo de *branch*



Nota: elaboración propia.

Figura N.º 49

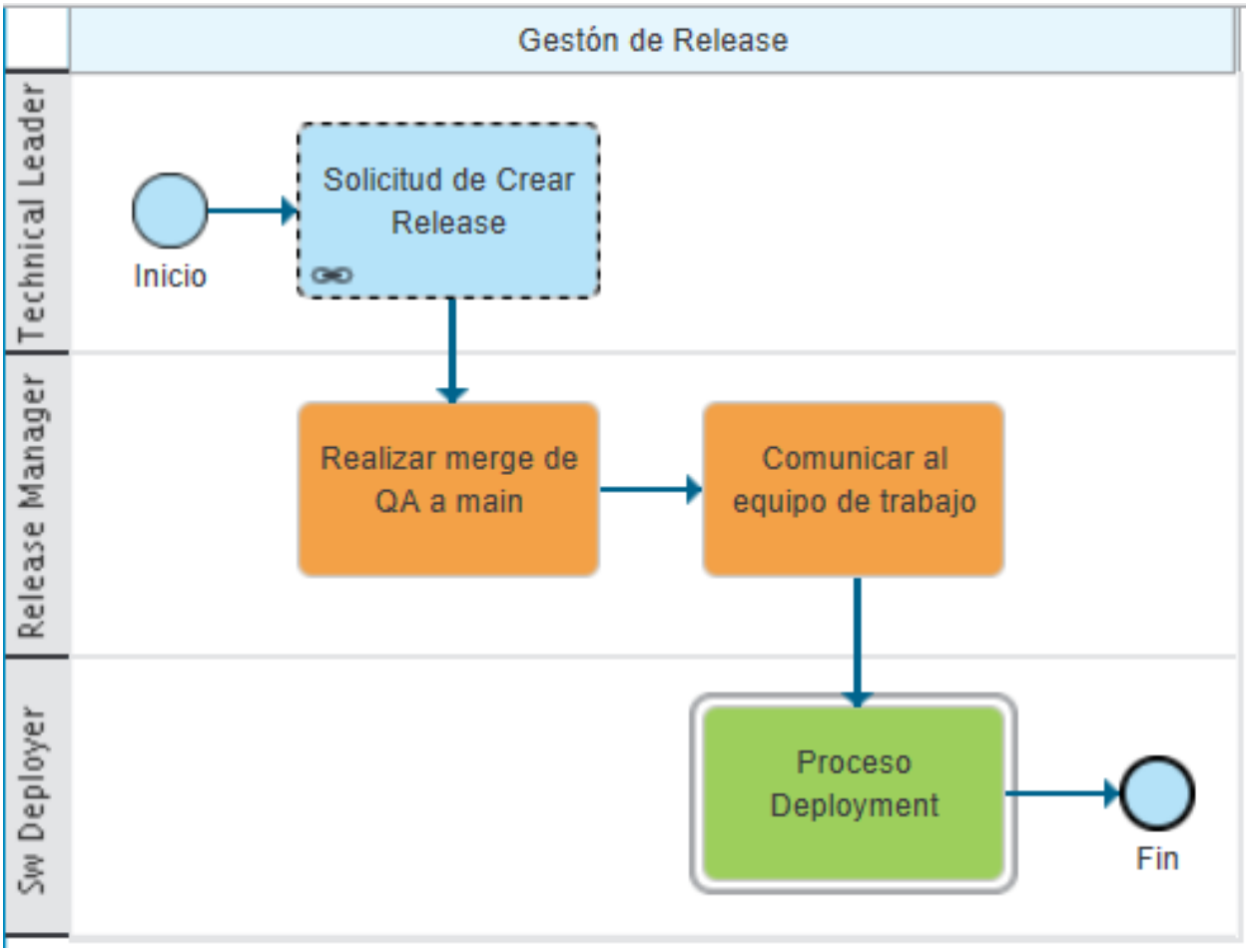
Diagrama de flujo subproceso gestión de *release* QA



Nota: elaboración propia.

Figura N.º 50

Diagrama de flujo subproceso gestión de *release*



Nota: elaboración propia.

Proceso de garantía del código.

Cuando existan diferencias de versión entre el cliente y GBM, y el proyecto se encuentra en garantía, se propone la realización de los siguientes pasos:

1. Documentar las diferencias encontradas y la razón de los cambios.

2. Se le envía la versión del cliente al *release manager* para que haga la gestión correspondiente de versionarla en el repositorio de GBM.

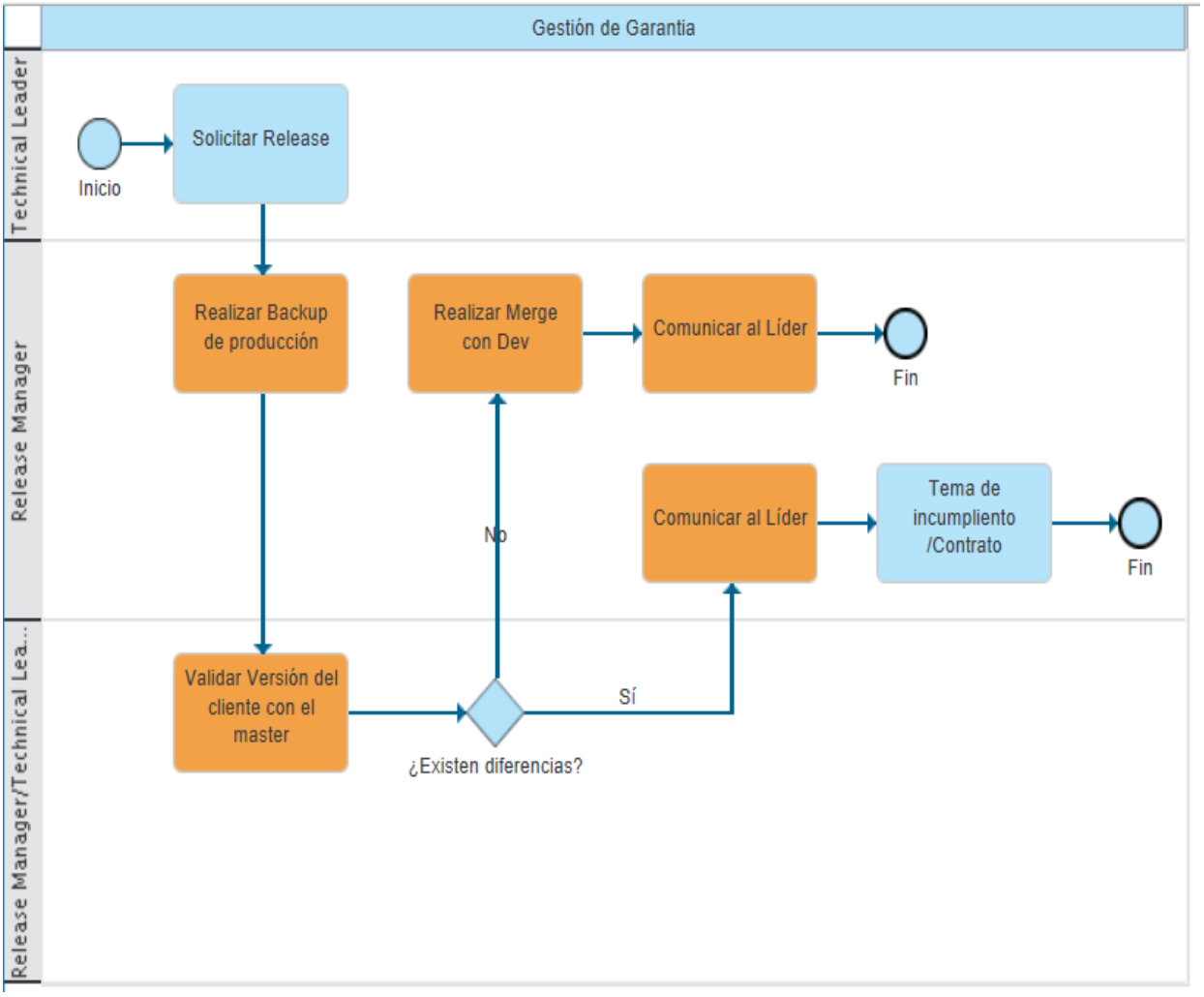
3. El *release manager* verifica la versión del cliente contra la versión de GBM, en caso de encontrar diferencias, se enviará un reporte comunicando lo encontrado.

• Si el proyecto está en garantía y existen estas diferencias, se le debe explicar al cliente que los cambios que están solicitando como garantía no se van a hacer efectivos, porque las versiones son diferentes y que existen cambios comparados con lo entregado inicialmente, es decir, el cliente realizó cambios al código en tiempos de garantía, por lo que, por aspectos contractuales, el mismo pierde el derecho a garantía.

A continuación, en la figura 51, se presenta el diagrama de flujo de la gestión de garantía propuesta.

Figura N.º 51

Diagrama de flujo subproceso gestión de garantía



Nota: elaboración propia.

Propuesta de recursos

Propuesta de roles

Los roles principales definidos en la metodología ágil para el área de Integration GBM son: equipo solucionador, Scrum máster y *product owner.* Teniendo en cuenta el contexto del área de estudio, se han definido otros roles que, por una parte, complementan el proceso de adopción de la agilidad y, por otra parte, soportan la gestión de los proyectos y la excelencia técnica en los desarrollos: ingeniero de calidad, *release manager*, analista, arquitecto de solución y líder técnico, quienes juntos conforman los OMT (Operation Management Team (OMT).

A continuación, se presenta el detalle operativo de cada uno de estos roles definidos:

Equipo solucionador.

Definición.

En el modelo ágil se prescribe que los equipos sean auto-organizados y cross-funcionales. En su interior se promueve el trabajo colaborativo, la transferencia de conocimiento y la comunicación entre los integrantes que lo conforman. Esto permite generar valor para el negocio a través de la solución adecuada y oportuna de necesidades del mismo. Asimismo, los equipos ágiles son estructurados y empoderados para organizar y gestionar su propio trabajo. La sinergia resultante optimiza la eficiencia del equipo.

Al equipo solucionador, que en lo posible estará compuesto por los mismos integrantes buscando mantener la cohesión, se le plantea la solución de diferentes necesidades. Sin embargo, se debe tener en cuenta que la especialización en algún campo de conocimiento también es aceptada, siempre y cuando el equipo, como un todo, tenga la capacidad de hacer que los elementos de trabajo se muevan con facilidad a lo largo de los estados de flujo sin generar dependencias ni cuellos de botella.

Es el equipo encargado de realizar la solución de cada elemento del *product backlog*, según el criterio determinado, se ha definido que el tamaño de cada equipo oscile entre 5 a 10 personas, teniendo en cuenta el personal con el que cuenta hoy en día el área en estudio y para lograr mayor cohesión, productividad y calidad, propiciar mejor entendimiento y desarrollo de capacidades y mejora continua, se considera como propuesta ideal que cada célula esté

compuesta por 10 personas, donde 7 estarán como equipo solucionador y 4 como miembros del OMT.

Considerando el contexto actual del área de Integration, sus relaciones con los proveedores y el grado de especialización que tienen actualmente, en el equipo solucionador, se tendrán que diferenciar dos perfiles: el de ingeniero desarrollador y el de ingeniero de calidad, aunque se espera que en la medida que el nivel de madurez en agilidad vaya aumentando, se vaya cerrando la brecha entre estos dos perfiles y se promueva la cross-funcionalidad entre desarrollo y pruebas.

Competencias y habilidades generales.

Las principales competencias y habilidades generales del rol son:

- Tener como prioridad hacer que pasen los elementos de trabajo entre los diferentes estados del flujo, respetando las limitaciones del inventario en proceso (WIP) y apoyándose entre los miembros del equipo en caso de ser necesario.
- Lograr ser un equipo cross-funcional y auto-organizado que sea capaz de solucionar las necesidades de negocio y asignarse a sí mismo las tareas, sin la necesidad de que un jefe o líder externo determine la forma en la que serán resueltos los problemas.
- Conseguir que todos y cada uno de los integrantes se sientan responsables por el cumplimiento del objetivo del *sprint*, de acuerdo con la planeación.
- Realizar todo lo que esté a su alcance para contribuir al éxito del equipo.
- Promover la transferencia de conocimiento en el interior de los equipos.
- Estar orientado a producir valor, todos los miembros del equipo ágil tienen que estar orientados a producir con calidad.
- Conocimiento de los marcos de trabajo ágiles y en especial de sus prácticas técnicas.
- Orientación al logro de objetivos.
- Capacidad de dividir el trabajo para identificar tareas que permitan el desarrollo de la solución, facilitar la visibilidad y el seguimiento del trabajo en proceso.

- Conocimiento técnico apropiado, de acuerdo con las tecnologías y plataformas.
- Actitud y capacidad para aprender, investigar y experimentar.
- Enfoque hacia la mejora continua personal, profesional y de los procesos.
- Capacidad de construir el diseño de *software*.
- Capacidades de comunicación asertiva.
- Capacidad de escuchar y de entender conceptos complejos.
- Disciplina para mantener la información del equipo y sus avances actualizada.
- Capacidad de diferenciar lo importante de lo urgente.

Competencias y habilidades específicas del perfil de desarrollo.

Las principales competencias y habilidades específicas del perfil de desarrollo son:

- Contar con conocimiento en desarrollo y pruebas de *software* técnicas.
- Conocimientos técnicos de lenguajes de programación.
- Un enfoque lógico y metódico del trabajo.
- Actitud y disposición para investigar y mantenerse al día con las nuevas tecnologías.
- Buenas habilidades escritas, para redactar informes, diagramas e instrucciones.
- Capacidad de análisis y resolución de problemas complejos.
- Paciencia y atención al detalle.
- Capacidad de trabajar bien por cuenta propia y como miembro de un equipo.
- Capacidad de afrontar problemas.
- Buenas habilidades de comunicación para prestar apoyo y asesoramiento a otros miembros del equipo.
- Actitud de servicio.
- Capacidad para concentrarse y mantener el foco.
- Capacidad para tomar la iniciativa.

- Conocimiento y habilidades para corregir defectos de *software*.
- Capacidad de análisis de código.
- Capacidad de diseñar y analizar de bases de datos.
- Capacidad de análisis de arquitectura.
- Habilidad para la programación.
- Habilidades interpersonales.
- Capacidad para hacer que los sistemas de *software* funcionen más eficazmente.
- Conocimientos para instalar y realizar pases entre ambientes.
- Actitud y habilidades para investigar, diseñar y desarrollar *software*.
- Capacidades para probar y mejorar el *software*.
- Disciplina para seguir los estándares, los lineamientos, buenas prácticas y para mantener la información de avance del trabajo actualizada.
- Trabajo bajo presión.

Competencias y habilidades específicas del perfil de calidad

Las principales competencias y habilidades específicas del perfil de calidad son:

- Capacidad de organización.
- Pensamiento lógico.
- Capacidad de planificar el trabajo futuro.
- Atención a los detalles.
- Habilidades de trabajo en equipo.
- Capacidad de estimar la duración de la construcción y pruebas en cada elemento del *product backlog.*
- Aptitudes para la comunicación verbal y escrita.
- Aptitudes para la elaboración de plan de pruebas y de diseño y ejecución de casos de pruebas.

- Capacidad para trabajar de forma organizada.
- Capacidad de análisis y resolución de problemas complejos.
- Capacidad para priorizar tareas.
- Capacidad de trabajar bajo presión.
- Capacidad de lectura y análisis de código fuente.
- Capacidad de análisis de diseño de base de datos.
- Capacidad de análisis de arquitectura.
- Conocimiento para elaborar y realizar pruebas sobre programas o sistemas ya construidos.
- Habilidad para evaluar.
- Habilidad en el desarrollo de casos pruebas automatizados que puedan repetirse y valorarse.
- Capacidad para identificar riesgos.
- Conocimiento en herramientas utilizadas para la realización y el registro de pruebas.
- Capacidad para recomendar formas de mejorar la realización de pruebas.
- Capacidad para resolver cómo probar riesgos a fin de que el *software* resulte lo más seguro posible.
- Disciplina para seguir los estándares, los lineamientos, buenas prácticas y para mantener la información de avance del trabajo actualizada.

Scrum máster.

Definición.

Es el facilitador de los equipos ágiles. Promueve la mejora continua y la correcta implementación de las prácticas del marco de trabajo, acompaña a los equipos en su día a día e identifica y remueve posibles impedimentos que surjan en el proyecto, con el propósito de que los equipos alcancen su nivel máximo de productividad.

El Scrum Máster es un rol que requiere dedicación exclusiva y se debe evitar que las personas que lo representen tengan asignaciones de trabajo diferentes; es fundamental que, además de la dedicación exclusiva, también sea estable en el tiempo, es decir, que los equipos a los que pertenezca también sean estables en el tiempo y se mantengan como un solo equipo.

Competencias y habilidades.

Las principales competencias y habilidades del rol son:

- Foco en asegurar que cada equipo esté llevando adecuadamente el marco de trabajo y mejorando continuamente.
- Amplio conocimiento en marcos de trabajo ágiles.
- Actitud y capacidad de aprender, investigar y experimentar.
- Amplias capacidades de facilitación.
- Capacidades de gestión.
- Capacidad de adaptación.
- Liderazgo servicial.
- Fuertes habilidades de comunicación verbal y escrita.
- Conocimientos básicos en gestión de proyectos.
- Conocimiento en conceptos básicos del área de TI.
- Conocimiento sobre el flujo entre arquitecturas de sistemas de información.
- Conocimiento general de las herramientas que se utilizan para el desarrollo de *software*.
- Conocimiento de la jerga de los equipos de desarrollo de *software*.
- Auto- organización.
- Habilidades para identificar, gestionar y resolver impedimentos.
- Habilidades de co-creación.
- Habilidades para identificar y propiciar la mejora continua de procesos o personas, mediante *coaching*, *mentoring* o consultoría según sea apropiado.

- Habilidades de liderazgo.
- Habilidad para resolver problemas complejos.
- Capacidad de identificar, manejar y resolver conflictos.
- Habilidades interpersonales.

Product owner.

Definición.

El *product owner* es el representante de todas las áreas de negocio involucradas en el problema de negocio que se pretende resolver. Se asegura de que el equipo entienda el valor que se pretende generar para el negocio con la ejecución del proyecto y con los incrementos de producto. En consecuencia, se cerciora de que el equipo comprenda el proyecto y su propósito e identifica qué funcionalidades son relevantes para su construcción. Debe estar disponible para el equipo durante todo el *sprint* y a lo largo de todo el proyecto, su dedicación no debe ser menor al 50 % de su tiempo.

Competencias y habilidades.

Las competencias de un *product owner* son los siguientes:

- Liderazgo servicial y para la toma de decisiones.
- Capacidad para velar por el mayor valor generado para el proyecto.
- Capacidad para decidir qué se construye y a qué se renuncia en el proyecto.
- Capacidad para velar por el cumplimiento del proyecto.
- Habilidades y disposición para evaluar constantemente prioridades del *product backlog*.
- Capacidad para control del inventario en proceso de cada uno de los equipos, de acuerdo con los estándares definidos para cada caso.
- Capacidades de facilitación.
- Capacidades de gestión.
- Habilidades de comunicación.

- Capacidad de adaptación.
- Conocimiento de información valiosa sobre los requerimientos actuales y funcionalidades a corto tiempo.
- Habilidad para identificar relaciones entre diferentes elementos, usada para la gestión de dependencias funcionales.
- Capacidad comunicativa importante, de tal forma que sea acreditado dentro de la organización que permite el fácil acceso a determinados grupos.
- Conocimiento en conceptos básicos del área de TI.
- Conocimiento del *roadmap* del negocio.
- Aptitudes para la comunicación verbal y escrita.
- Aptitudes para la escucha activa.
- Aptitudes para redactar y entender historias de usuario.
- Capacidad para identificar relaciones e interacciones entre diferentes sistemas e identificar servicios necesarios.
- Capacidad de análisis.
- Capacidad para entablar buenas relaciones con la gente.
- Capacidad para plantear preguntas con claridad.
- Capacidad para trabajar bajo presión.

Ingeniero de calidad.

Definición.

Se responsabiliza de certificar la calidad de las aplicaciones de *software* que desarrolle. Con ese fin establece una estrategia de pruebas para los proyectos.

Competencias y habilidades

Las principales competencias y habilidades del rol son:

- Capacidades para realizar análisis amplios sobre necesidades del negocio y los requerimientos técnicos asociados a las necesidades.

- Habilidades para llevar el control de la ejecución de los procesos y lineamientos establecidos para garantizar la estimación de tareas, recursos, tiempos y costos asociados al proceso de certificación.
- Habilidades para la ejecución de planes de pruebas
- Capacidad de adaptación para replantear el plan de pruebas y garantizar su cobertura de acuerdo a las necesidades.
- Habilidades para asegurar que se ejecuten los procesos y lineamientos establecidos en el ámbito de calidad.
- Habilidades para garantizar la gestión de defectos identificados que se produzcan en los diferentes ciclos de pruebas que son aplicados a una solución completa o a una necesidad específica de negocio.
- Habilidades de facilitación para la ejecución de los procesos y lineamientos establecidos para asegurar el adecuado diseño y ejecución de los procesos de pruebas automatizadas y de desempeño que se realicen en una solución completa o en una necesidad específica de negocio.
- Gobierno en control de la calidad.
- Conocimiento de los procesos de pruebas.
- Conocimiento y capacidades de automatización de pruebas.
- Flexibilidad y capacidad de adaptación.
- Actitud y capacidad de aprender, investigar y experimentar
- Capacidades de facilitación.
- Capacidades de gestión.
- Capacidad de observación y atención al detalle.
- Liderazgo servicial.
- Fuertes habilidades de comunicación verbal y escrita.
- Conocimientos básicos en gestión de proyectos.

- Conocimiento en conceptos básicos del área .
- Conocimiento básico sobre el flujo entre arquitecturas de sistemas de información.
- Conocimiento general de las herramientas que se utilizan para el desarrollo de *software*.
- Conocimiento de la jerga de los equipos de desarrollo de *software*.
- Auto- organización.
- Habilidades para identificar y propiciar la mejora continua.
- Habilidad para resolver problemas complejos.
- Capacidad de identificar, manejar y resolver conflictos.
- Habilidades interpersonales.

Analista de solución.

Definición.

Es el responsable de entender las necesidades de negocio, a partir de los lineamientos y prioridades establecidas por el dueño de producto. Se encarga principalmente de asegurar que las necesidades identificadas y priorizadas se conviertan en elementos de la lista de necesidades y que estas cumplan con las características definidas por la organización, así como de asegurar el entendimiento de las necesidades por parte de los equipos.

Competencias y habilidades.

- Capacidad de entender las necesidades de negocio y de plasmarla adecuadamente en las historias de usuario y demás elementos del *product backlog.*
- Claridad y capacidad de comunicar el entendimiento de las necesidades de negocio a los equipos solucionadores.
- Foco en asegurar de que cada equipo solucionador, esté trabajando en los elementos más prioritarios, de acuerdo a lo definido por el dueño de producto.
- Capacidades de facilitación.
- Capacidades de gestión.

- Habilidades de comunicación.
- Capacidad de adaptación.
- Liderazgo servicial.
- Conocimiento de información valiosa sobre las necesidades actuales y funcionalidades a corto tiempo.
- Habilidad para identificar relaciones entre diferentes elementos, usada para la gestión de dependencias funcionales.
- Capacidad comunicativa importante, de tal forma que sea acreditado dentro de la organización que permite el fácil acceso a determinados grupos.
- Conocimiento en conceptos básicos del área de TI.
- Conocimiento sobre de flujo entre arquitecturas de sistemas de información
- Conocimiento general de las herramientas que se utilizan para el desarrollo de *software*
- Conocimiento de la jerga de los equipos de desarrollo de *software*
- Capacidad de análisis de necesidades en *software*.
- Aptitudes para la comunicación verbal y escrita.
- Aptitudes para la escucha activa.
- Aptitudes para redactar y entender historias de usuario.
- Capacidad para identificar decisiones operacionales y para facilitar la definición reglas de negocio.
- Capacidad para identificar, desde el conocimiento del negocio, relaciones e interacciones entre diferentes sistemas.
- Capacidad de análisis.
- Capaz de dar información compleja de un modo directo.
- Capaz de entablar buenas relaciones con la gente.
- Capaz de plantear preguntas con claridad.

- Capaz de trabajar bajo presión.
- Capaz de trabajar con vencimientos.
- Conocimientos especializados del negocio.
- Creatividad.
- Tacto, diplomacia y buenas habilidades de negociación.
- Buenas habilidades de redacción de historias de usuario.
- Habilidad para resolver problemas complejos.
- Capacidad de investigación de problemas de negocio.
- Capacidad de identificar y proponer mejoras, tanto a los productos, como a los procesos.
- Trabajo en equipo.
- Conocimiento del roadmap del negocio.

Líder técnico.

Definición.

Persona experta, encargada del apoyo técnico y de conocimiento del negocio al equipo solucionador.

Competencias y habilidades.

Las principales competencias y habilidades del rol son:

- Capacidad de entender las necesidades de negocio desde la perspectiva técnica y de plasmarla adecuadamente en las historias de usuario y demás elementos del *product backlog.*
- Claridad y capacidad de comunicar el entendimiento de las necesidades técnicas al equipo solucionador.
- Conocimiento experto en plataformas y tecnologías relacionadas con los aplicativos asociados al proyecto.
- Actitud y capacidad de investigación de nuevas y mejores prácticas y tecnologías.

- Capacidades de facilitación.
- Capacidades de gestión.
- Habilidades de comunicación.
- Capacidad de adaptación.
- Liderazgo servicial.
- Capacidad para diferenciar lo importante de lo urgente.
- Fuertes habilidades en desarrollo de *software*.
- Habilidades básicas de arquitectura de *software*.
- Conocimiento de técnicas de análisis y resolución de problemas técnicos.
- Habilidad para identificar relaciones entre diferentes elementos, usada para la gestión de dependencias técnicas.
- Capacidad para identificar decisiones operacionales y definir reglas de negocio.
- Capacidad para identificar relaciones e interacciones entre diferentes sistemas e identificar servicios necesarios.
- Orientación a los estándares técnicos.
- Capacidad para tomar decisiones.
- Flexibilidad y capacidad de adaptación.
- Capacidad de planificación y priorización
- Capacidad de mantener la calma bajo presión.
- Habilidades de comunicación verbal y escrita.
- Capacidad de influenciar y motivar a los demás.
- Trabajo en equipo.
- Aptitudes para la escucha.
- Capacidad de analizar problemas complejos y encontrar soluciones.
- Capacidad de evaluar códigos de *software* con lineamientos y estándares.

- Capacidad de evaluar y minimizar riesgos técnicos en el desarrollo del *software*.
- Capacidad de enseñar y formar técnicamente al equipo solucionador.
- Habilidad para negociar y llegar a acuerdos técnicos con el equipo solucionador.
- Habilidades interpersonales.

Release manager.

Definición.

Este rol se debe encargar de gestionar y asegurar temas de versionamiento y ramificación de los proyectos de la fábrica de *software*.

Competencias y habilidades

Las principales competencias y habilidades del rol son:

- Esta persona debe tener un conocimiento general de cada aspecto del desarrollo del ciclo de vida del *software*.
- Excelentes habilidades de comunicación con los diferentes equipos de desarrollo.

Integration Management Team (IMT).

Definición.

El IMT es el equipo que representa la figura de dueño de producto en los equipos Scrumban para la fábrica de *software*. Es el encargado de gestionar y priorizar las historias de usuario o necesidades que son atendidas por la fábrica, así como la capacidad de las células y sus equipos.

Los integrantes del equipo IMT son:

- Arquitecto de solución.
- Scrum máster.
- Gestor de portafolio.
- Analista de negocio.

Competencias y habilidades.

Las principales competencias y habilidades generales en el equipo son:

- Conocimiento global del negocio, del estado de las aplicaciones y las prioridades organizacionales.
- Empoderamiento para la toma de decisiones con respecto a la evolución de los productos existentes y del manejo de los incidentes, de acuerdo con las prioridades organizacionales.
- Capacidad para generar información oportuna y de calidad, con el fin de informar a los otros participantes sobre lo que los equipos de Scrumban están desarrollando.
- Responsabilidad sobre el éxito de la solución que se está desarrollando o mejorando, sin importar que el producto sea una aplicación interna o un producto externo.
- Enfoque en comunicar y hacer entender la importancia de entregar el mayor valor posible, lo cual incluye la atención continua en la excelencia técnica y la calidad.
- Priorización de las necesidades y manejo de conflictos de capacidades.
- Gobierno en arquitectura.
- Gobierno en negocio.
- Habilidades de facilitación.
- Liderazgo servicial.
- Habilidades de gestión.
- Comunicación asertiva.
- Habilidades de negociación y manejo de conflictos.

Operation Management Team (OMT).

Definición.

El OMT es el equipo que funge como facilitador de los equipos solucionadores que conforman una célula Scrumban, aseguran que el trabajo de la célula se esté ejecutando lo mejor

posible, con un servicio eficiente que cumpla con las expectativas y necesidades de los interesados

Cada célula cuenta con un OMT dedicado a los equipos solucionadores que la conforman.

Los integrantes del OMT pueden ser:

- Líderes técnicos.
- Arquitectos de solución.
- Scrum másters.
- Analistas de negocio
- *Product owners.*
- Analistas.
- Ingeniero de calidad.

Competencias y habilidades.

Las principales competencias y habilidades generales en el equipo son:

- Capacidad de entender las necesidades de negocio y de plasmarla adecuadamente en las historias de usuario y demás elementos del inventario de necesidades.
- Claridad y capacidad de comunicar el entendimiento de las necesidades de negocio a los equipos solucionadores.
- Foco en asegurar que cada equipo que conforma la célula esté trabajando en los elementos más prioritarios para el negocio.
- Control del inventario en proceso de cada uno de los equipos, de acuerdo a los estándares definidos para cada caso.
- Gobierno en arquitectura en el contexto de la célula.
- Gobierno en negocio en el contexto de la célula.
- Capacidades de facilitación.
- Capacidades de gestión.

- Habilidades de comunicación.
- Capacidad de adaptación.
- Liderazgo servicial.

Artefactos

Los marcos ágiles cuentan con artefactos (plantillas) que representan el trabajo o valor en diversas formas, los cuales son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos en los marcos ágiles están diseñados específicamente para maximizar la transparencia de la información clave, que es necesaria para asegurar que todos tengan el mismo entendimiento del artefacto.

Para efectos de la gestión de proyectos, se definen artefactos que apoyen al proceso de la fábrica para producir entregables que generen valor a sus clientes. En el modelo Scrumban se propone una serie de artefactos que permiten tener seguimiento y control al momento de la construcción de historias de usuario en el mantenimiento de las aplicaciones, o bien, al momento de solucionar los incidentes de soporte.

A continuación, se describen los artefactos necesarios para la implementación de modelo Scrumban:

Inventario de necesidades o *product backlog.*

Lista de elementos que describen las necesidades del negocio, ordenadas según el valor que generan y que establece el dueño de producto o el equipo administrador del producto, se visualiza en la herramienta de gestión de requerimientos y gestión del proyecto.

Inventario de necesidades del *sprint* o *sprint backlog.*

Describe y contiene la lista de tareas que un equipo ágil elabora en la reunión de planificación de la iteración, como plan para completar los objetivos o conjunto de necesidades seleccionados para la iteración y que se compromete a demostrar al cliente al finalizar la iteración en forma de incremento de producto preparado para ser entregado, se visualiza en la herramienta de la gestión del proyecto.

Historia de usuario.

La historia de usuario debe representar la necesidad del usuario, en un lenguaje común y conversacional que, con la orientación del dueño de producto, debe ser escrita y explicada por el analista de negocio.

Debe describir qué se desea, pero no cómo se debe realizar, es decir, está enfocada en entender la necesidad y no en definir la solución. Cada funcionalidad completada a partir de la implementación de la historia de usuario genera valor para el negocio de manera iterativa e incremental. Contiene toda la información necesaria para su desarrollo.

Se debe tener en cuenta que la condición de cada historia de usuario es que el aporte del producto resultante sea una funcionalidad completa. Las historias de usuario que no contengan una funcionalidad completa no generan valor, ya que no aportan al producto de manera iterativa e incremental.

El formato propuesto para la documentación de las historias de usuario requiere del registro de la siguiente información:

Línea de producto: se especifica la línea de producto a la que corresponde la solución de la necesidad.

Módulo: se especifica el módulo donde es requerida la solución.

Nombre de historia de usuario: se registra el nombre asignado a la necesidad.

Tipo: se anota el tipo de solución requerida, ya sea proyecto o mantenimiento.

Nombre del proyecto: se detalla el nombre del proyecto.

ID de la historia de usuario: se asigna el ID de la HU.

Prioridad de valor de negocio: se registra la prioridad dada por el dueño del producto según el valor que la solución de la necesidad le brindará al negocio, se registra bajo los niveles alto, medio, bajo.

Estimación: se registra una estimación del esfuerzo en puntos requerido para el desarrollo de la historia de usuario; este debe reflejar el consenso del equipo, tanto desde el punto de vista de desarrollo como el de pruebas.

Dependencia: se registran las dependencias con otras historias de usuario, se digita el ID de las historias de usuario correspondientes.

Usuario (yo como): se registra el rol de la persona que requiere de la solución y, por ende, quién define la necesidad.

Necesidad (quiero / necesito): se registra la necesidad comunicada por el dueño del producto o usuario experto.

Valor o beneficio (para qué): se registra el valor o beneficio que la solución de esta necesidad traerá al negocio.

Criterios de aceptación: Se registran las expectativas del dueño del producto sobre la solución dada a la necesidad (Apéndice 4).

Check list calidad.

Se utiliza tanto en el proceso de preventa como en el refinamiento de las historias de usuario, tiene como objetivo guiar al ingeniero de calidad durante el proceso para el cumplimiento de sus responsabilidades asegurando a la vez que toda solicitud que llegue a los equipos solucionadores vaya con la información necesaria para poder trabajar sin interrupciones (Apéndice 5).

Matriz de dependencias.

Para la construcción de la matriz de dependencias, se toma como base los insumos identificados en eventos anteriores: Inception y mapeo de historias de usuario. La matriz es almacenada en la herramienta de trabajo y es actualizada permanentemente durante el desarrollo de las necesidades.

Contiene los siguientes elementos:

- Nombre del elemento o dependencia.
- Área o a quien pertenece el elemento o dependencia.
- Área o persona que gestiona el elemento o dependencia.
- Persona que soluciona el elemento o la dependencia.
- Impacto.
- Fecha posible solución.
- Dependencias con otros elementos. (Apéndice 6)

Sprint Review.

Artefacto que presenta los resultados de la sesión de revisión de la iteración, incluyendo la cantidad de historias de usuario que fueron desarrolladas, los mayores inconvenientes y riesgos presentados durante la iteración, así como cualquier solicitud de mejora o cambio que el *product owner* comunique al equipo.

De encontrarse algún defecto durante la presentación de las funcionalidades, el mismo es documentado en este artefacto para su seguimiento. El encargado de la documentación de este artefacto es el Scrum másster del equipo. (Apéndice 7)

Sprint Retrospective.

Artefacto donde se registran los resultados de la sesión de retrospectiva, se documenta la explicación de la técnica utilizada, el registro fotográfico de la misma.

El objetivo de la sesión es lograr que el equipo piense y comunique todo aquello que creen salió bien durante la iteración, todo lo que debe mejorar y todo aquello que no debe volver a ocurrir. El encargado de la documentación de este artefacto es el Scrum máster del equipo. (Apéndice 8)

Sprint Dashboard.

Artefacto que documenta las principales métricas de los resultados de la iteración contemplando las prácticas de requerimientos, calidad y *release management*, acá se reporta la velocidad del equipo y la productividad del mismo durante la iteración. El encargado de la documentación de este artefacto es el Scrum máster del equipo (Apéndice 9).

Plan de *release.*

Se documenta al inicio del proyecto, el *release manager* basado en el mínimo producto viable y en la priorización del trabajo por parte del *product owner* documenta el plan de *releases* a producción (Apéndice 10).

Solicitud de *release management.*

La solicitud de creación de repositorio se debe realizar cuando un nuevo proyecto requiera de un repositorio para almacenar el código fuente que se vaya a generar. El encargado de realizar esta solicitud es el líder técnico que conforma el OMT de la célula (Apéndice 11).

Solicitud de versionamiento.

Los equipos de desarrollo deben solicitar que se realice el proceso de *release* por ejemplo:

• Cuando ya se tienen muchas funcionalidades listas y el equipo de desarrollo considere que se necesite realizar un *release* del desarrollo.

• Cuando ya existe una versión y el equipo de desarrollo está trabajando en nuevas funcionalidades o corrección de incidentes.

• Cuando QA realice las pruebas requeridas y estás cumplen con satisfacción se requiere que soliciten el *release* a producción. (Apéndice 12)

Release notes.

Los equipos de la célula deben realizar el *release notes* cada vez que se entregue una nueva versión del producto. El encargado de completar este documento es el líder técnico del OMT junto con el equipo que conforma la célula. Debe llevar al menos la siguiente información:

Fecha: en la que se genera la nueva versión.

Número de *release*: número de *release* de la versión (ejemplo: 1.0, 2.1).

Funcionalidades agregadas/modificadas: nombrar las funcionalidades nuevas o modificadas que tiene la nueva versión.

Descripción: una descripción lo más representativa posible de lo que se está entregando.

Bugs cerrados: en caso de que aplique, indicar cuáles *bugs* se están cerrando en la nueva versión que se está entregando (Apéndice 13).

Plan de calidad.

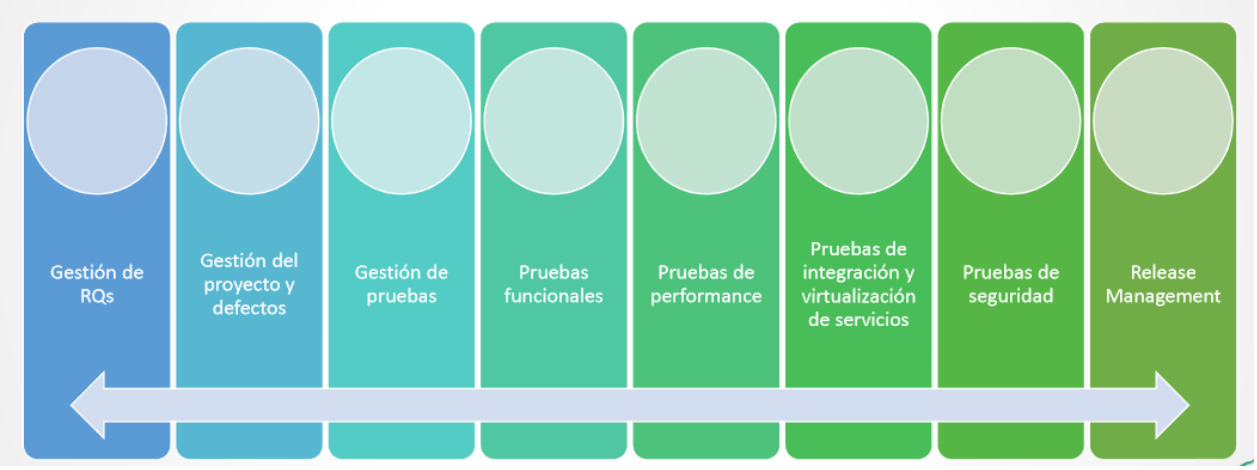
Es documentado por el ingeniero de calidad del OMT al inicio del proyecto, incluye el alcance que tendrán las pruebas, los recursos, los entregables, los criterios de aceptación y rechazo por los cuales los *testers* se regirán y el plan de trabajo y organización respectiva. (Apéndice 14)

Propuesta de herramientas colaborativas (Tool Chain)

Una vez definida la estructura de los equipos y la metodología basada en prácticas ágiles, es necesario definir las herramientas colaborativas que mejor se adapten al esquema propuesto desde que inicia hasta que finalizan los desarrollos, de la siguiente manera:

Figura N.º 52

Ciclo de desarrollo de la fábrica



Nota: elaboración propia.

Una vez realizado un análisis comparativo de herramientas colaborativas (Apéndice 15), se determina que existen tres ecosistemas de gestión y producción posibles para la implementación del esquema metodológico propuesto, a continuación, se presenta el análisis de las tres propuestas:

1. Ecosistema IBM (puro)

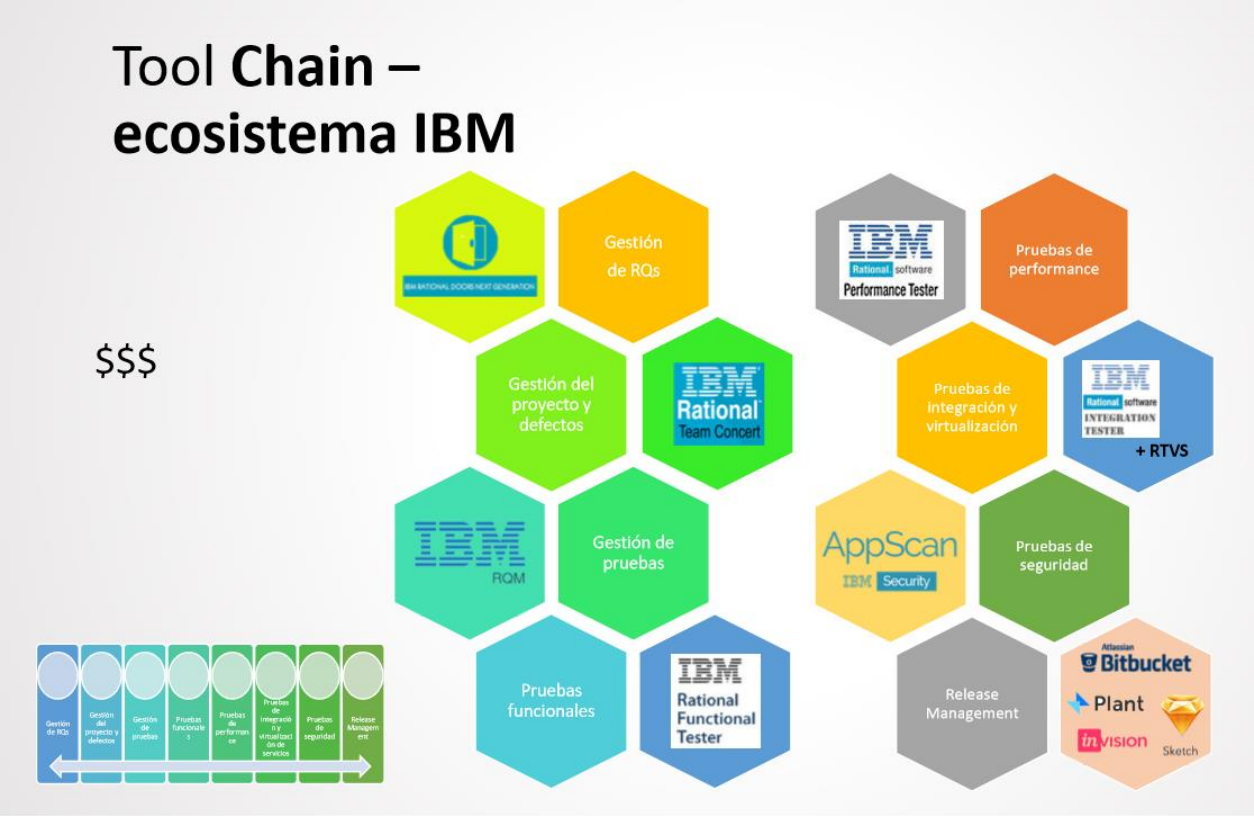
En este ecosistema, se cuenta con todas las herramientas de un mismo proveedor, lo que permite transparencia en las integraciones de todos los componentes que se utilizan durante la gestión. Esta opción presenta un alto costo de licenciamiento comparado con las otras, tanto de los *tools* especializados para *testing* como de las licencias requeridas para los roles de gestión (Ver detalle en la propuesta económica).

En este caso puntual del área de Integration, las herramientas son *on premise*, lo que significa un incremento adicional en almacenamiento y gestión de las mismas. Las herramientas

que conforman el ecosistema #1 para la Gestión de requerimientos, Gestión del proyecto, Gestión de la calidad y *release management* son las mostradas en la figura 53.

Figura N.º 53

Tool chain Ecosistema IBM



Nota: elaboración propia

2. Ecosistema Open (puro)

En este escenario, se cuenta igualmente con todo un *tool chain* para gestión y *testing* especializado. Este ecosistema se compone de una serie de herramientas compradas separadamente en modalidad *on cloud*. Los *test cases* (casos de prueba) y toda la información de gestión de la calidad permanece almacenada en la herramienta Practitest (QA Manager open).

El *testing* especializado (automatización funcional, rendimiento) se realiza con herramientas *open* sin costo, lo cual abarata el costo por licenciamiento. Sin embargo, requerimientos que incluyan escenarios especiales de automatización funcional (arquitecturas de *testing* centralizado) y rendimiento (con muchos usuarios virtuales más allá de la capacidad de las herramientas *open*) no podrían ser solventados.

Esta opción excluye el testing especializado de seguridad, integración y virtualización, ya que solo con herramientas IBM puede llevarse a cabo.

En cuanto a gestión de requerimientos y del proyecto en sí (Jira/ Confluence), se puede contar con información fluida, con la excepción del manejo de identificadores de las historias de usuario o requerimientos, ya que la herramienta Confluence lo gestiona mediante la generación automática en Jira. Esto implica una capacitación de los usuarios para ingresar los requerimientos en Confluence, de manera que la información sea fácil de consultar posterior a su ingreso.

Otro aspecto importante es que, para consultar los avances de las historias de usuario o requerimientos en la herramienta Confluence, se requiere también licencia de Jira, es decir, estas dos herramientas son dependientes. En caso de usuarios remotos como lo serían los clientes, es requerido crear o configurar una pantalla de consulta que permita únicamente la consulta de estatus sin entrar al detalle de la información de gestión (data interna GBM).

Las herramientas que conforman este ecosistema se detallan a continuación en la figura 54.

Figura N.º 54

Tool Chain Ecosistema Open

Nota: elaboración propia.

3. Ecosistema Mixto

El ecosistema mixto tiene como objetivo armar un ecosistema tipo Lego, donde se permita la combinación de herramientas IBM con herramientas *open source*, de forma que entre ambas y presentando escenarios más económicos, se logre cubrir la necesidad del esquema metodológico presentado. A continuación, se muestran en la tabla 21 las opciones.

Tabla N.º 21

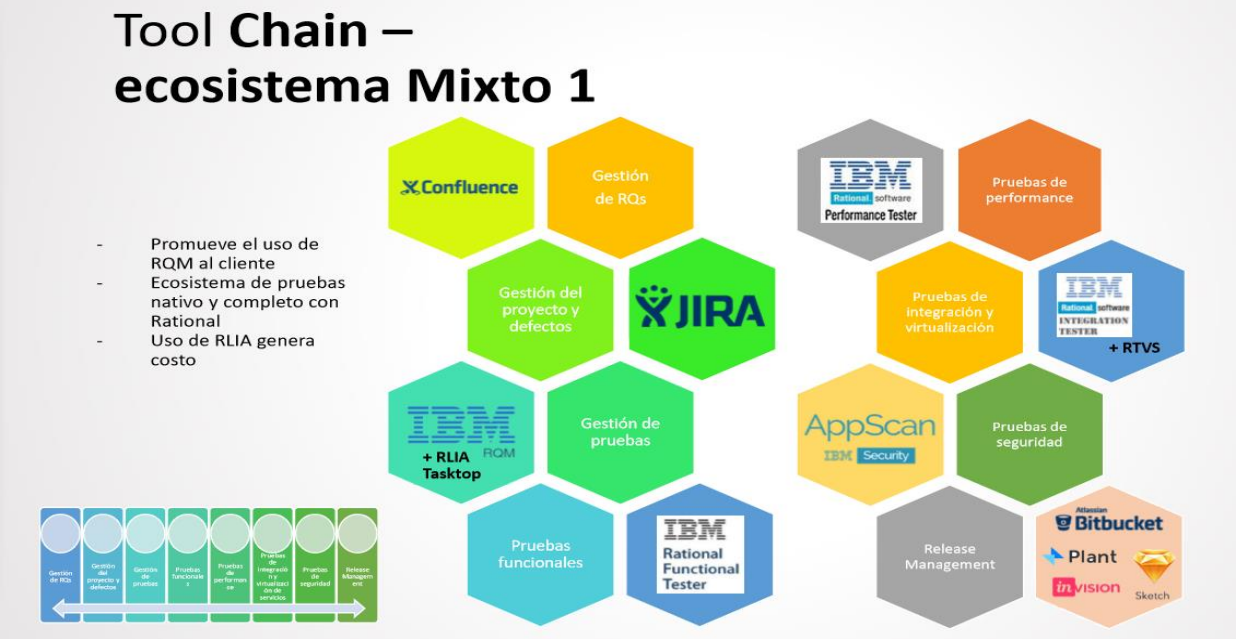
Ecosistema Mixto opción 1

Opción	Detalle de composición de herramientas.	Pros	Contras
Opción 1	[Jira+ Confluence +plug in+ RTC (para uso de QA) + Quality Manager] + Testing especializado con IBM.	Ahorro en licenciamiento IBM del equipo solucionador (developers, SM, PM, otros).	Impacto de trazabilidad de la información de QA debido al uso del plug in, (costo $592 por procesador) por año.

Nota: elaboración propia.

Figura N.º 55

Tool Chain Mixto 1



Nota: elaboración propia.

Tabla N.º 22

Opción 2 Ecosistema Mixto

Opción 2	[Jira+ Confluence + Practitest] + Testing especializado con IBM.	Ahorro en licencias IBM de gestión.	Retrabajo en calidad, se lleva manualmente el control entre casos de prueba, gestión y ejecución de *testing* especializado.

Nota: elaboración propia.

Figura N.º 56

Tool Chain Ecosistema Mixto Opción 2

Nota: elaboración propia.

Métricas de los equipos Scrumban

Definición de objetivos

Al definir que la prioridad de la fábrica es la satisfacción del cliente mediante la entrega temprana y frecuente de soluciones con valor y que esta satisfacción se produce a partir del trabajo responsable de los equipos que mejoran continuamente, se plantean entonces para los equipos solucionadores en Scrumban, los siguientes objetivos:

- Mantener un flujo continuo de atención de necesidades.
- Atender las necesidades de mantenimiento y soporte oportunamente y con la calidad esperada.
- Mantener el foco en la mejora continua.

Definición de indicadores

Metas para el cumplimiento de los indicadores

Definiciones

Para cada indicador definido se establece:

- **Un valor objetivo**: nivel de servicio esperado para cada indicador, que es la base para el cálculo del componente por calidad del servicio.
- **Un umbral mínimo**: nivel de servicio mínimo permitido, cuyo incumplimiento tiene consecuencias contractuales.
- **Una ponderación para cada indicador**: cada uno de los indicadores tendrá una ponderación que formará parte de una calificación de los equipos solucionadores. Se proponen las siguientes ponderaciones de acuerdo con el valor que representan en las prácticas ágiles de desarrollo de *software*.

Con el cálculo de cada indicador y su ponderación relativa, se calcula entonces un nivel de servicio general del equipo solucionador.

Tabla N.º 23

Propuesta de objetivos para las células en Scrumban

Indicador	Ponderación del indicador	Métricas	Umbral mínimo	Objetivo *iteración* 1 y 2	Objetivo
Continuidad en el flujo	30	Porcentaje de elementos con más de 5 días en proceso	Por debajo de 10%	5%	0%
Tiempo de ciclo	30	Porcentaje cumplimiento acuerdos de tiempo de ciclo	Por encima de 85%.	90%	100%
Efectividad de la mejora	40	Porcentaje de cumplimiento de efectividad de mejora	Por encima de 85%.	90%	100%

Nota: elaboración propia.

1. **Continuidad en el flujo de necesidades.**

Es un indicador mensual que mide la relación entre el número de elementos del inventario de necesidades por resolver que han estado priorizadas en *product backlog* de la célula, sin haber sido asignadas a los equipos solucionadores por más de cinco días hábiles y el número total de necesidades priorizadas en el mismo período de tiempo.

Tabla N.º 24

Continuidad en el flujo de necesidades

	Continuidad en el flujo de necesidades
Definición	La continuidad en el flujo de necesidades está representada en la comparación entre la cantidad de necesidades por resolver que han estado en proceso por más de cinco días hábiles y la cantidad total de necesidades por resolver en proceso
Fórmula	*Continuidad en el flujo = ((NP5 / NPT) * 100%)* *NP5 = cantidad de necesidades que han estado en proceso por más de 5 días hábiles.* *NPT = cantidad total de necesidades en proceso.*
Procedimiento	Al final de cada *sprint* o iteración, se determina la cantidad total de necesidades que se tienen en proceso y de esa cantidad, se determina cuántas necesidades han estado en proceso por más de cinco días. Luego se calcula la relación entre las dos. Una vez obtenidos los resultados, se presenta en un informe al IMT para su valoración y se convierte en el insumo para la realización de los diferentes seguimientos entre el OMT y el equipo. De encontrarse oportunidades de mejora, se deben realizar planes de acción conjuntamente con el equipo solucionador para implementarlos y promover la mejora continua.
Periodicidad	Cada iteración Scrumban (dos semanas).
Meta	5 % para las primeras dos iteraciones y 0 % para el resto de las iteraciones.
Herramienta	Herramienta para la gestión del proyecto.
Responsable de la medición	Analista OMT.

Nota: elaboración propia.

2. Tiempo de ciclo.

Mide la relación entre el tiempo total de ciclo desde que una necesidad entra a la fábrica, hasta que es finalizada por el equipo solucionador. Para efectos de cálculo, se separa y se pondera por cada tipo de necesidad.

Tabla N.º 25

Indicador tiempo de ciclo

	Tiempo de ciclo
Definición	El indicador de tiempo de ciclo pretende comparar, basado en el tiempo total de ciclo, el tiempo real contra el tiempo establecido en los acuerdos de servicio.
Fórmula	*Tiempo de ciclo = Promedio de ((TCC / TCR) * 100%)* *TCC = Tiempo de ciclo comprometido que corresponde al acuerdo de servicio.* *TCR = Tiempo de ciclo real.*
Procedimiento	Para cada necesidad atendida durante una iteración de Scrumban, se determina el tiempo de ciclo total y se compara contra su respectivo acuerdo de servicio, dependiendo del tipo de necesidad. Al final de la iteración, se obtiene un promedio del cumplimiento de los acuerdos de servicio. Una vez obtenidos los resultados, se presenta en un informe al IMT para su valoración y se convierte en el insumo para la realización de los diferentes seguimientos entre el OMT y el equipo. De encontrarse oportunidades de mejora, se deben realizar planes de acción conjuntamente con el equipo solucionador para implementarlos y promover la mejora continua.
Periodicidad	Cada iteración Scrumban (dos semanas).
Meta	90 % para las primeras dos iteraciones y 100 % para el resto de las iteraciones.
Herramienta	Herramienta para la gestión del proyecto.
Responsable de la medición	Analista OMT.

Nota: elaboración propia.

3. Efectividad de la mejora continua.

Mide mensualmente la relación entre los compromisos de efectividad de mejora continua y los resultados correspondientes en un equipo en los últimos tres meses. La efectividad de la mejora continua es evidenciada a partir de la disminución de incidentes relacionados con las causas que originaron las oportunidades de mejora.

Tabla N.º 26

Indicador efectividad de la mejora continua

	Efectividad de la mejora continua
Definición	Mide mensualmente la relación entre los compromisos de efectividad de mejora continua y los resultados correspondientes en un equipo en los últimos tres meses. La efectividad de la mejora continua es evidenciada a partir de la disminución de incidentes relacionados con las causas que originaron las oportunidades de mejora.
Fórmula	*Efectividad de mejora = ((MCR / MCC) * 100%)* *MCR = Mejora continua real. Cantidad real de mejoras efectivas.* *NPT = Mejora continua comprometida. Cantidad comprometida de mejoras efectivas.*
Procedimiento	Cada mes se determina qué historias de usuario de mejora fueron efectivas, en términos de si tuvo o no impacto en la cantidad de incidentes del tipo que generó la historia de mejora continua en la aplicación correspondiente. Para calcular el indicador, se toma la información mensual de los tres meses anteriores al cálculo Los compromisos de mejora continua son definidos cada mes por cada equipo junto con su respectivo OMT. Una vez obtenidos los resultados, se presenta en un informe al IMT para su valoración y se convierte en el insumo para la realización de los diferentes seguimientos entre el OMT y el equipo. De encontrarse oportunidades de mejora, se deben realizar planes de acción conjuntamente con el equipo solucionador para implementarlos y promover la mejora continua.

Periodicidad	Mensual
Meta	90 % para las primeras dos iteraciones y 100 % para el resto de las iteraciones.
Herramienta	Herramienta para la gestión del proyecto.
Responsable de la medición	Analista OMT.

Nota: elaboración propia.

Adicionalmente, se definen las siguientes métricas como resultado de las prácticas ágiles:

Tabla N.º 27

Métricas prácticas ágiles

Área	Target	Métrica	Aplicable en	Benchmark de Industria	Significado	Aporte de valor	Formula
RQ's	DEV	% de solicitudes de cambio por solicitante				Permite enfocar esfuerzos en los procesos que estén generando más cambios del requerimiento durante su desarrollo	
	QA	% Solicitudes de cambio no aprobados					
RM	DEV	Total de Commits por Sprint	Al realizar check-in	No definido	Respeta las reglas establecidas release management	Asegura el cumplimiento de mejores prácticas de administración del codigo.	
	DEV	Commits por desarollador por Sprint	Al realizar check-in	No definido	Respeta las reglas establecidas release management	Asegura el cumplimiento de mejores prácticas de administración del codigo.	
	Release	Cantidad de Release Entregados					
Metodología	DEV	Sprint Burndown	Durante el desarrollo del sprint	100% del cumplimiento del compromiso del sprint	Cuanto tiempo le falta al equipo para completar el trabajo en relación al compromiso que se estableció en el Sprint	Análisis de tiempo para finalizar el compromiso del sprint	
		Team Velocity	Al finalizar el sprint durante el sprint review	100% de velocidad	Velocidad de desarrollo del equipo en términos de puntos	Indica el rendimiento del equipo	
		Cantidad de HU comprometidas vrs completadas	Al finalizar el sprint	no definido	Velocidad de desarrollo del equipo en términos de puntos	Indica el rendimiento del equipo	
UX	UX	% de solicitudes de cambio de diseño por solicitante					
		% Solicitudes de diseños no aprobados					
Calidad	QA	Efectividad de la prueba	Al finalizar QA	>= 70%	Indica el valor aportado de las pruebas.	Permite identificar los gaps de mejora para el equipo de Testing sobre la gravedad de defectos.	((((+HB) + (+CB) + (+MB)) * 100) / NDT
		Defectos rechazados	Al finalizar QA	<= 20%	Indica si la labor de análisis del Tester fue adecuada.	Permite identificar los gaps de análisis de los Testers.	(+RB * 100) / NDT

Nota: elaboración propia.

Buenas prácticas ágiles por implementar

Técnicas de estimación

Pivote.

¿Qué es un pivote?

Es un artefacto definido de tal forma que permite determinar todos aquellos elementos necesarios para dar una estimación lo más certera posible basados en una base lógica de estimación, toma en cuenta los esfuerzos mínimos de una labor específica.

¿Cuál es el objetivo del pivote?

Colaborar en la estimación de esfuerzo de las actividades de desarrollo y calidad de historias de usuario.

¿Qué elementos lo conforman?

Está conformado por las siguientes partes:

1. Una secuencia Fibonacci establecida para la medición por puntos en relación con horas.
2. Una estimación detallada de una historia de usuario que contemple todas las posibles actividades de desarrollo y calidad.
3. La estimación en términos de horas para las actividades de desarrollo y calidad de la historia de usuario completa

¿Quién define el pivote?

Lo construye el Scrum Team, o bien, líderes técnicos o expertos en la materia.

¿Cuándo define el pivote?

Se construye antes del refinamiento.

¿Cómo se define un pivote?

Se reúnen los miembros del equipo y se selecciona una historia de usuario que contemple todas las funciones que se podrían desarrollar en una historia de usuario. Dicha historia de usuario se debe desglosar de tal forma que se puedan determinar aquellas fases que se necesitan para desarrollarla, así como las actividades y subactividades de dichas fases.

Por ejemplo teniendo esta historia de usuario:

"Yo como dueño del producto deseo un módulo de mantenimiento que me permita actualizar la información de los clientes del banco para tener una base de datos actualizada".

Se analiza la historia de usuario y según el conceso del equipo, se determina que existen tres fases para su desarrollo:

1. *Entendimiento de la historia de usuario*
2. *Desarrollo de la historia de usuario*

Luego, si se requiere un poco más de detalle, se pueden definir subetapas y actividades, de tal forma que se logre mejorar el entendimiento de la historia de usuario y en consecuencia su estimación.

Planning poker.

La técnica de *planning poker* permite hacer una estimación inicial del proyecto rápida y fiable, dado que todos los miembros del equipo comparten sus diferentes informaciones y expresan su opinión sin sentirse condicionados por el resto. Cada número significa un peso, esfuerzo o complejidad para completar un objetivo (historia de usuario). La numeración de las cartas está basada en la sucesión de Fibonacci. La distancia entre números crece conforme se hacen mayores. De esta manera, se facilita la decisión sobre qué tamaño tiene un objetivo.

¿Es un 5 o un 8? No vale la pena entretenerse en pensar si es un 5 o un 6, ni en el error por no intentar buscar esta precisión, dado que se compensará por encima y por debajo con el resto de las estimaciones.

Planning poker es un proceso iterativo de planificación. Funciona de la siguiente manera:

- El cliente lee un objetivo (historia de usuario escrita en una tarjeta).
- El equipo le hace preguntas para entender su alcance.
- Las respuestas importantes se pueden apuntar en la propia tarjeta como detalles del objetivo o condiciones.
- Pueden aparecer nuevas historias de usuario.
- Cada miembro del equipo piensa en el esfuerzo necesario para completar el objetivo y todos muestran sus tarjetas simultáneamente, de manera que no están condicionados por las estimaciones de los otros.

- Las personas que están más alejadas del consenso explican por qué su votación es más alta (hay algún problema en el que nadie más ha pensado o el resto no ha tenido en suficiente consideración) o más baja (conocen una manera sencilla de resolver el problema, resolvieron algo muy parecido en un proyecto anterior, etc.).
- El equipo vuelve a votar, hasta que alcanza un acuerdo. No hay democracia, dado que todos deberán comprometerse a que ese objetivo se va a acabar con el esfuerzo acordado (se supone que no hay una persona que siempre vota de manera singular y a la que nunca se puede convencer).

Diversas personas manifestaron que con esta técnica el equipo ha ido mejorando mucho la precisión de sus estimaciones. De hecho, existen diversos estudios que concluyen que la sinergia que produce esta estimación conjunta es mucho mejor que la de una única persona por separado (típicamente la del jefe de proyecto o un *senior* en un proyecto tradicional).

Algunos consejos y trucos:

- Elegir un objetivo de tamaño típico en el proyecto como patrón con el cual comparar el resto de los objetivos y asignarle una puntuación de carta también media (por ejemplo, un 5).
- Puede ser conveniente no jugar con las cartas más altas, ya que el error de estimación es mucho mayor (además, objetivos tan grandes dificultan ver el progreso y se tarda más en hacer visible si existen problemas en el proyecto). Notar que en la fotografía de las barajas se han apartado las cartas más altas para no jugar con ellas.
- Dada la dificultad de empezar a trabajar con puntos de historia y no con esfuerzo en días ideales, para facilitar empezar con el uso del Planning Poker, se puede hacer el símil de que, por ejemplo, dos puntos de historia corresponden a un día ideal.
- Comparar el tamaño que se va dando a cada objetivo respecto al de otros (triangulación).

Estimación por camisetas.

La estimación por afinidad es una técnica utilizada para estimar HU en un tiempo corto, es decir, para generar una estimación general previa a la estimación de la planificación.

Para esto se requiere:

1. Lista de *product backlog* o historias de usuario.
2. Acuerdo de medidas de estimación, que para estos efectos serán las **tallas de camisetas (S, M, L).**
3. Un tablero que contenga en las columnas las medidas de la métrica por utilizar con el objetivo de asociar cada HU a una medida específica.

Primero, se inicia con la explicación a los miembros de la sesión del proyecto, luego cada miembro recibe N cantidad de HU, las cuales debe leer y estimar según la métrica establecida.

Peer review o revisión de pares

El objetivo de ejecutar la revisión de pares o Peer review es evaluar y garantizar la calidad del código fuente que se desarrolla en el equipo, así como mantener la integridad del código. Por esta razón, se propone crear un lineamiento que colabore y controle la ejecución de la revisión de pares.

A continuación, se propone el siguiente lineamiento para Peer review:

- Bajo cada historia de usuario en la herramienta utilizada para la gestión de los desarrollos para el cual se desarrolla la funcionalidad, se debe crear la tarea Peer review.
- Se debe asignar la tarea Peer review a un miembro del equipo contemplando que la tarea debe ser realizada por otro miembro distinto al que desarrolla el requerimiento.
- Cuando el desarrollo del requerimiento asociado se finaliza, se inicia con la ejecución de la tarea de Peer review.
- El miembro del equipo asignado a la ejecución de la tarea de peer review ejecuta la revisión correspondiente.
- Las observaciones que se encuentren durante la revisión deben ser comunicadas al miembro de equipo que desarrolló el requerimiento, para que solucione según las recomendaciones dadas por el ejecutor de la revisión.
- Una vez que se atienden las solicitudes u observaciones, la tarea relacionada al Peer review debe pasar a estado *Done.*

El encargado de asignar la ejecución de la tarea de Peer review es el líder técnico del OMT, ello debido a su conocimiento tanto del equipo como del área técnica relacionada. Esta asignación se realiza durante la ejecución de la iteración, cada vez que sea necesaria una Peer

review. El líder técnico, cada vez que termina el desarrollo de historia de usuario, le indica a uno de los miembros del equipo solucionador que ejecute el Peer review.

Se propone la utilización de la plantilla Peer review (Apéndice 16) para el registro, de tal forma que funcione como una guía para la ejecución de la práctica. Esta guía debe ser utilizada cada vez que se necesite ejecutar un Peer review. Si el desarrollador lo desea, puede adjuntar la guía utilizada a la tarea de Peer review. Sin embargo, no es un proceso que se tenga que considerar obligatorio y queda a elección del equipo, su implementación de forma obligatoria o condicionada según la madurez de este.

Implementación de la propuesta

Plan de implementación

El plan de implementación supone la formulación de los planes de acción para poder implementar la propuesta. Dada la similitud de la operatividad entre las células, se propone la implementación de un plan piloto con una duración de tres meses, que incorpore únicamente una de las células, con el fin de validar la propuesta, ajustar lo necesario y replicar seguidamente al resto de la fábrica.

La célula seleccionada para el plan piloto, según disposición de la jefatura de la fábrica, es la célula especializada en BPELS. El plan detalla las actividades por realizar, los responsables, la fecha de inicio y la duración de esta actividad (Apéndice 17).

Tabla N.º 28

Plan de ejecución piloto

Nombre de tarea	Duración	Trabajo	Comienzo	Fin
Plan Piloto Fábrica de integration	**89,5 días**	**6 538 horas**	**mié 2/5/18**	**mar 4/9/18**
Diseño de propuesta	**50 días**	**474 horas**	**mié 2/5/18**	**mar 10/7/18**
Planeación	**3 días**	**72 horas**	**mié 2/5/18**	**vie 4/5/18**
Preparación para capacitación	**24,5 días**	**128 horas**	**lun 7/5/18**	**vie 8/6/18**
Ejecución de capacitaciones	**1 día**	**96 horas**	**vie 8/6/18**	**lun 11/6/18**
Ejecución del Piloto	**61 días**	**5 768 horas**	**lun 11/6/18**	**mar 4/9/18**

Nota: elaboración propia.

Beneficios esperados

La empresa GBM debe evaluar la factibilidad de implementar el cambio metodológico en el área de Integration, por lo que debe encontrar argumentos sólidos traducidos en beneficios que obtendrá si realiza la inversión necesaria para el cambio.

Los argumentos pueden tener una connotación financiera, operativa, técnica u organizacional y en efecto, la factibilidad puede ser evaluada desde diversos campos: financiero, operativo, técnico, comercial, organizacional, legal, entre otros.

Con la implementación del plan piloto, se pretende evidenciar los beneficios estratégicos del cambio metodológico y procedimental, dado que se considera que la principal oportunidad de mejora que enfrenta el área de Integration es el establecimiento de una metodología de trabajo y una definición de procesos que engloben una forma de trabajo estandarizada, de forma tal que su productividad mejore y a la vez los coloque en una posición más competitiva a nivel del mercado.

Es importante entender que es a partir del mes dos o de la tercera iteración del plan piloto donde se empezarán a percibir los beneficios acá descritos, ya que el modelo requiere de un lapso para estabilizarse y alcanzar una madurez necesaria para correr de forma satisfactoria. Se presenta, a continuación, en la tabla 29, un análisis de los beneficios esperados con la implementación de la metodología propuesta, así como las acciones necesarias para obtenerlos.

Tabla N.º 29

Beneficios esperados

Beneficios metodologías ágiles	¿Cómo se obtienen?
Gestión regular de las expectativas del cliente. El cliente establece sus expectativas indicando el valor que le aporta cada requisito del proyecto y cuando espera que esté completado.	Lista de requisitos priorizada. El cliente crea y gestiona la lista de requisitos del producto o proyecto, donde quedan reflejadas sus expectativas a nivel de requisitos, valor, costo y entregas.
Cumplimiento de expectativas El cliente comprueba de manera regular si se van cumpliendo sus expectativas, da retroalimentación, ya desde el inicio del proyecto puede tomar decisiones informadas a partir de resultados objetivos y dirige estos resultados del proyecto, iteración a iteración, hacia su meta. Se ahorra esfuerzo y tiempo al evitar hipótesis.	Demostración de los resultados de proyecto en cada iteración. Al final de cada iteración el equipo demuestra al cliente los requisitos que ha conseguido completar. Tras una inspección del resultado real del proyecto hasta ese momento y considerando el esfuerzo que ha sido necesario para realizarlo, el cliente solicita los cambios que necesita y replanifica el proyecto.
Resultados anticipados (*time to market*)	Priorización de requisitos por valor y costo

El cliente puede empezar a utilizar los resultados más importantes del proyecto antes de que esté finalizado por completo. Siguiendo la ley de Pareto (el 20% del esfuerzo proporciona el 80% del valor), el cliente puede empezar antes a recuperar su inversión (o autofinanciarse), comenzando a utilizar un producto al que solo le faltan características poco relevantes, puede sacar al mercado un producto antes que su competidor, puede hacer frente a urgencias o nuevas peticiones de clientes, etc.	Al inicio de cada iteración el cliente prioriza la lista de requisitos del producto o proyecto en función del valor que le aportan, su costo de desarrollo y los riesgos del proyecto, cambiando los requisitos previstos para reaccionar a cambios de contexto en el proyecto. El progreso del proyecto se mide en función de los requisitos que el equipo completa en cada iteración.
Flexibilidad y adaptación De manera regular, el cliente redirige el proyecto en función de sus nuevas prioridades, de los cambios en el mercado, de los requisitos completados que le permiten entender mejor el producto, de la velocidad real de desarrollo, etc. Al final de cada iteración, el cliente puede aprovechar la parte de producto completada hasta ese momento para hacer pruebas de concepto con usuarios o consumidores y tomar decisiones en función del resultado obtenido.	Replanificación en el inicio de cada iteración Se asume que los cambios son parte natural del proyecto. Toda iteración comienza con una replanificación del proyecto. Esta replanificación no es traumática, puesto que Scrum minimiza el número de objetivos/requisitos en los que el equipo trabaja (WIP, Work In Progress) a los que caben en una iteración. Todavía no se ha hecho ningún esfuerzo por desarrollar los requisitos de las siguientes iteraciones. El hecho de que los requisitos se completen en función del valor que aportan al cliente minimiza la probabilidad de que se produzcan grandes cambios en el transcurso del proyecto.
Retorno de inversión (ROI) De manera regular, el cliente maximiza el ROI del proyecto. Cuando el beneficio pendiente de obtener es menor que el coste de desarrollo, el cliente puede finalizar el proyecto.	Priorización de requisitos por valor Cada iteración el cliente dispone de unos requisitos completados y replanifica el proyecto en función del valor que le aportan los requisitos pendientes respecto del costo de desarrollo que tienen.

Mitigación de riesgos Desde la primera iteración, el equipo tiene que gestionar los problemas que pueden aparecer en una entrega del proyecto. Al hacer patentes estos riesgos, es posible iniciar su mitigación de manera anticipada. "Si hay que equivocarse o fallar, mejor hacerlo lo antes posible". El *feedback* temprano permite ahorrar esfuerzo y tiempo en errores técnicos. La cantidad de riesgo a la que se enfrenta el equipo está limitada a los requisitos que se pueden desarrollar en una iteración. La complejidad y riesgos del proyecto se dividen de manera natural en iteraciones.	Desarrollo iterativo e incremental Un requisito se debe completar en una iteración. El equipo debe realizar todas las tareas necesarias para completarlo y que esté preparado para ser entregado al cliente con el esfuerzo mínimo necesario. De esta manera, no se deja para el final del proyecto ninguna actividad arriesgada relacionada con la entrega de requisitos.
Productividad y calidad De manera regular, el equipo va mejorando y simplificando su forma de trabajar.	Mejora continua En cada iteración, el equipo realiza una retrospectiva, para analizar su manera de trabajar e identificar los obstáculos que le impiden avanzar al mejor ritmo posible.
Los miembros del equipo sincronizan su trabajo diariamente y se ayudan a resolver los problemas que pueden impedir conseguir el objetivo de la iteración. La comunicación y la adaptación a las diferentes necesidades entre los miembros del equipo son máximas (se van ajustando iteración a iteración), de manera que no se realizan tareas innecesarias y se evitan ineficiencias.	Comunicación diaria del equipo Todo miembro del equipo conoce cómo el trabajo de los otros miembros impacta en el suyo y cuáles son las necesidades de los otros.
Las personas trabajan más enfocadas y de manera más eficiente cuando hay una fecha límite a corto plazo para entregar un resultado al que se han	*Time boxing* Cada iteración siempre tiene la misma duración (1 mes, 4 horas, 10 día etc.), por lo que las personas

comprometido. La consciencia de esta limitación temporal favorece la priorización de las tareas y fuerza la toma de decisiones. Las iteraciones son regulares para facilitar la sincronización sistemática con otros equipos, con el resto de la empresa y con el cliente.	aprenden lo que pueden conseguir en este tiempo, cómo organizarse, priorizar tareas y tomar decisiones.
El equipo minimiza su dependencia de personas externas para poder avanzar (depender de la disponibilidad de otros puede parar tareas).	Equipo multidisciplinario El equipo está formado por todas las personas con las especialidades necesarias para llevar a cabo el proyecto.
La estimación de esfuerzo y la optimización de tareas para completar un requisito es mejor si la realizan las personas que van a desarrollar el requisito, dadas sus diferentes especializaciones, experiencias y puntos de vista. Así mismo, con iteraciones cortas la precisión de las estimaciones aumenta.	Estimación de esfuerzo conjunta En el inicio de la iteración, los miembros del equipo estiman de manera conjunta el esfuerzo necesario para completar requisitos y sus tareas.
Las personas trabajan de manera más eficiente y con más calidad cuando ellas mismas se han comprometido a entregar un resultado en un momento determinado y deciden cómo hacerlo, no cuando se les ha asignado una tarea e indicado el tiempo necesario para realizarla	Compromiso del equipo En el inicio de cada iteración, el equipo selecciona los requisitos que se compromete a completar y entregar al final de la iteración (responsabilidad). El propio equipo se organiza (autoridad) identificando las tareas necesarias, su esfuerzo y auto asignándose cada miembro las tareas que se compromete a realizar.
El equipo se evita caminar mucho tiempo por un camino equivocado que le obligue a realizar un gran esfuerzo para llegar al objetivo esperado.	Demostración de resultados preparados para ser utilizados y velocidad sostenida. Por un lado, al final de cada iteración, el equipo

Se asegura la calidad del producto de manera sistemática y objetiva, a nivel de satisfacción del cliente, requisitos listos para ser utilizados y calidad interna del producto.	demuestra al cliente los requisitos que ha conseguido completar, de manera que están completamente operativos. Por otro lado, para tener una velocidad de desarrollo sostenida, el equipo necesita desarrollar cada incremento de producto sin tener que revisitar aspectos mal resueltos en iteraciones anteriores.
Alineamiento entre cliente y equipo Los resultados y esfuerzos del proyecto se miden en forma de objetivos y requisitos entregados al negocio. Todos los participantes en el proyecto conocen cuál es el objetivo por conseguir. El producto se enriquece con las aportaciones de todos.	Cliente y equipo trabajando “en equipo” En cada iteración, el equipo y el cliente trabajan juntos en la creación de los requisitos del proyecto (en la estimación de la lista priorizada de requisitos del proyecto), en darles detalle (en la reunión de planificación de la iteración) y en el análisis del resultado obtenido (en la demostración de los requisitos completados).
Equipo motivado Las personas están más motivadas cuando pueden usar su creatividad para resolver problemas y cuando pueden decidir organizar su trabajo.	Equipo autogestionado El equipo es quien se compromete a completar unos requisitos determinados en una iteración y quien mejor sabe cómo desarrollarlos. Por ello es el equipo quien se autoorganiza y quien planifica cómo trabajará en la iteración.
Las personas se sienten más satisfechas cuando pueden mostrar los logros que consiguen.	Demostración: en cada iteración, el equipo muestra al cliente los resultados que consigue. No está meses trabajando sin poder exhibir su obra.

Nota: elaboración propia.

No cabe duda de que todos estos factores a primera vista son positivos y conllevan el buen funcionamiento de una empresa y, por tanto, parece lógico que traigan beneficios económicos. Sin embargo, en la forma tradicional de llevar los proyectos, se comete el error de centrarse demasiado en el volumen de ventas y el tamaño de la compañía, en vez de en los beneficios que el proyecto en sí pueda traer.

Se puede estar convencido de que todas las buenas ideas proporcionadas por metodologías ágiles proporcionarán éxito, pero se considera imprescindible buscar, no solo vender (que claramente es bueno), sino el beneficio económico, que es la materialización de tal éxito. Hay que tener claro que el mayor costo de la implementación de la presente propuesta no es el de una inversión económica (aunque a simple vista parece indicar lo contrario), sino que viene determinado por los esfuerzos y la capacidad para organizar y gestionar correctamente la adopción de esta metodología, que es realmente el mayor reto por venir.

En este sentido, se propone el inicio de la implementación de una *Economía creativa*, la cual es considerada como el paradigma necesario en las empresas del nuevo siglo para obtener una alta flexibilidad a los cambios y adaptación a las turbulencias del mercado, productividad, generación de innovación y altos dividendos, y en lo que está basada la agilidad. Cualquier empresa que no transforme su modelo y siga tomando sus decisiones con base en paradigmas tradicionales, tendrá dificultades para competir, un alto nivel de desperdicio en sus procesos y ofrecerá productos menos atractivos. GBM lo tiene claro al día de hoy y de ahí surge la necesidad de la presente investigación.

Análisis Costo -Beneficio

Por el tipo de inversión que requiere la propuesta de cambio metodológico y procedimental, se selecciona el análisis costo -beneficio como apoyo a la gerencia para la decisión de implementar o no los cambios propuestos.

Para efectos del análisis Costo-Beneficio y en acuerdo con el área de Integration, se decide evaluar los beneficios que conllevaría el cambio metodológico comparado con la inversión necesaria para poder implementar la propuesta, donde los beneficios por considerar son: la reducción del tiempo de ciclo del desarrollo de los requerimientos y la eliminación de una cantidad considerable de horas de uno de los roles más costosos y con mayores oportunidades de mejora de la forma de trabajo actual, el rol del *project manager.*

Como parte de esta economía creativa, se encuentran casos de éxito y estadísticas que demuestran que, mediante las prácticas ágiles de desarrollo de *software* y habiendo alcanzado los equipos una madurez considerable, el ciclo de desarrollo desde la solicitud hasta su entrega puede disminuirse de un 33 % hasta un 62 % al año. Para efectos de la presente propuesta, se fija en conjunto con la gerencia el beneficio esperado en un 10%, una vez finalizado el plan piloto.

Esto se logra como un resultado integral de dichos estudios, donde se plasma la posibilidad de lograr la disminución de retrabajos en márgenes desde el 10 % al 40%, la reducción de tiempos de análisis y planificación en un 10 %, aumento en la gestión de proyectos y control sobre el proceso de desarrollo de *software* desde un 10 % hasta inclusive un 90 % (Index, BTM Business Agility, 2010; PMI, 2017; CHAOS, 2015). Se fija en 10 % tomando en cuenta que la empresa GBM es una empresa conservadora, muy burocrática y estructurada, por lo que el cambio, si bien es posible, puede presentar limitaciones de aceptación del mismo.

Además, los estudios presentan como una oportunidad para las empresas que inicien su cambio hacia prácticas ágiles, el lograr hasta un 30 % de ventaja en rendimiento en crecimiento de ingresos y ganancias (PMI, 2017), una disminución del costo del ciclo de desarrollo de un requerimiento en un 42 % y la posibilidad de tener el retorno de la inversión en un 37 % más rápido, así como el aumento de la satisfacción del cliente en un 49 % (Resultados publicados de encuesta a 403 organizaciones que han adoptado principalmente ágil), beneficios que se esperan alcanzar una vez esté toda la fábrica con el modelo replicado.

Agregado al beneficio de la reducción del tiempo de ciclo, se acuerda priorizar como segundo gran beneficio la reducción de horas para el rol de *project manager*, uno de los roles más costosos en la metodología actual y del cual el cliente y a nivel interno GBM ha presentado insatisfacciones. El beneficio obtenido al darse la eliminación del costo de mantener el rol de un PM en un 60 % será tangible desde el mes número uno del piloto, mientras que, para poder iniciar la visualización del beneficio de la reducción del tiempo de ciclo, se requerirá al menos de un mes (dos iteraciones), es decir, el alcance de esta meta es gradual a través de las primeras iteraciones, mientras la metodología madura en conjunto con los equipos.

Es importante recalcar que, actualmente, el área en estudio no cuenta con registros formales o históricos robustos que demuestren su productividad real, tiempos de respuesta por etapa ni métricas actuales, sin embargo, con el fin de realizar el análisis costo beneficio manualmente, se extrae información de correos y se toma como base la capacidad actual del equipo seleccionado. Se presenta, a continuación en la tabla 30, el análisis realizado para poder representar de forma monetaria los beneficios esperados, siendo esto de $ 537 024.

Tabla N.º 30

Beneficios esperados

Criterio	Racional	Integration-GBM metodología actual	Integration GBM-Metodología ágil	Beneficio $	# Meses	Total
Reducción Tiempo ciclo desarrollo	Posibilidad de reducción al final del piloto de la duración del tiempo de ciclo de un requerimiento en un 10% (Involucra aumento de la cantidad de requerimientos con una mejor distribución de habilidades en roles , mejoras en la calidad logrando disminución de retrabajos,disminución del tiempo de gestión, planeación y análisis,aumento de la velocidad de los recursos)	116160	255552	255552	2	511104
Rol Project Manager	Reducción del costo rol de Project Manager($45/hora/160 mes/3 meses)	21600	12960	8640	3	25920
Total						**537024**

Nota: elaboración propia.

Costos de la propuesta.

El área de Integration debe considerar entre los gastos para la implementación del plan piloto aspectos como la duración del piloto, costos de mano de obra para la preparación y ejecución de las capacitaciones, los materiales por utilizar en las mismas, así como el costo de la mano de obra durante la implementación de este.

Este análisis se hace tomando el salario por hora de un ingeniero estándar valorado en $66 por hora, en jornadas donde los empleados laboran 8 horas al día y tomando como referencia el costo del dólar a la fecha 25/03/2017 de ₡567,34, según el Banco Central. Para una valoración total. Se estiman los siguientes datos:

Costo de mano de obra.

El costo de la mano de obra necesario para el piloto se refiere al costo por hora del consultor que ha realizado la presente propuesta, el costo por hora de los *coaches* de las diferentes prácticas ágiles (Scrum Coach, Requirements Coach, Quality Coach y Release Management coach) que guiarán el plan piloto y las horas de los desarrolladores durante la ejecución del piloto, sin embargo, el costo correspondiente a los desarrolladores es el mismo costo que hoy en día cubre la empresa y este representa el monto más alto del costo de mano de obra. Es requerida la capacitación de todos los recursos que conformarán el IMT de la fábrica y la célula seleccionada.

La capacitación se divide en dos partes: la primera es una capacitación metodológica, la cual debe contener tanto la explicación de la nueva estructura como la explicación de los nuevos procesos, las prácticas ágiles, los roles y artefactos. Y la segunda parte debe ser una capacitación técnica que enseñe a los recursos cómo, cuándo y para qué se utilizarán las herramientas colaborativas seleccionadas. Cada *coach* será el facilitador de la capacitación tanto teórica como técnica de su práctica ágil. El costo de la mano de obra considerando la duración del piloto invertido en preparación y ejecución del mismo es de $ 468 668,81 .

Tabla N.º 31

Mano de obra plan piloto

Recurso	Cantidad de horas	Costo por Hora	Cantidad de recursos	Costo sin Cargas Sociales	% Cargas Sociales	Total	Costo total
Consultor propuesta	474	60	1	28440	50,33	14313,85	42753,85
Scrum Coach	180	60	1	10800	50,33	5435,64	16235,64
Requirements Coach	172	60	1	10320	50,33	5194,06	15514,06
QA Coach	172	60	1	10320	50,33	5194,06	15514,06
Release Manager Coach	172	60	1	10320	50,33	5194,06	15514,06
Desarrolladores/QAs (capacitación y ejecución del piloto)	488	45	11	241560	50,33	121577,15	363137,15
Total	1658			311760	50,33	156908,81	468668,81

Nota: elaboración propia.

Costo de materiales de capacitación.

Adicionalmente, se incorpora el costo de los materiales que se conforma de los artículos que se necesitan para la preparación y ejecución de las capacitaciones, considerando la participación de 11 recursos pertenecientes a la célula. Estos materiales están valorizados en un costo aproximado de $ 134,95.

Tabla N.º 32

Lista de materiales necesarios capacitación plan piloto

Material	Cantidad	Detalle	Costo/unidad	Costo total
Papel de construcción	2	Block (rojo, rosado, amarillo, beige, café, blanco, celeste, azul)	3290	6580
Pliego papel periódico	11	Pliegos	300	3300
Goma	4	Pequeñas	1440	5760
Tijeras	4		1990	7960
Regla	4	30 cm	490	1960
Cartulina	1	Blanca	6290	6290
Marcadores	12	Negros	490	5880
Post its	5	Blocks multicolor	5990	29950
Marshmallows	2	Bolsa tamaño mediano el marshmellow	2000	4000
Spaguettis	2	Bolsa grande de spaguettis delgados	1250	2500
Masking tape	2	Delgado	840	1680
Suma total				75860

Suma total dólares		Costo del dólar	562,15	134,95

Nota: elaboración propia.

Costo de herramientas colaborativas.

Se presenta a continuación el costo por escenario correspondiente al conjunto de herramientas colaborativas propuesto para la implementación de la metodología, para cada escenario se tomó en cuenta lo siguiente:

- Piloto con duración tres meses
- Herramientas IBM por dos meses (un mes de laboratorio gratuito)
- Once licencias para gestión de desarrollo
- Diez licencias para gestión de requerimientos
- Tres licencias para *testing open source*
- Dos licencias testing especializado IBM
- Ocho licencias para versionamiento de código
- Ocho licencias para prototipado
- Un *plug-in* para conectar herramienta IBM con *open source*

Costo Ecosistema IBM.

Si se decide realizar el piloto con el Ecosistema IBM, no se incurrirá en costo alguno correspondiente a herramientas colaborativas durante el primer mes, ya que existe la posibilidad de realizar el piloto durante un mes como un laboratorio, al cual la célula escogida del área de Integration podrá tener acceso. Esto como beneficio que recibe IBM al ser representante oficial en la región de IBM.

El costo de este Ecosistema incluye, entonces, los costos de las licencias IBM por dos meses más el costo de las licencias *open* de diseño por tres meses, siendo el total de la inversión $ 44 953. A continuación, en la tabla 33 se muestra el desglose.

Tabla N.º 33

Costos escenario Ecosistema IBM

Ecosistema IBM	Meses	
Licencias	1	2
IBM Team Concert (11 usuarios/$159/mes)	1750	3500
Doors On cloud (11 usuarios/$164/mes)	1804	3608
Rational funtional tester(2 licencias / mes)	1278	2556
Quality test manager (2 licencias / mes)	318	636
IBM performance tester (2 licencias / mes)	375	750
IBM Rational Integration tester	2160	4320
Appscan	29000	29000
Total	36685	44370

Nota: elaboración propia.

Tabla N.º 34

Costo total Ecosistema IBM

Costo total Ecosistema IBM	
Ecosistema IBM	44370
Licencia Bitbucket	150
Licencia Guthub	216
Licencia Plant	144
Licencia Sketch	48
Licencia Invision	25
Total	44953,00

Nota: elaboración propia.

Costo Ecosistema Open Puro.

Tal como se observó en secciones anteriores de la presente investigación, el Ecosistema Open está conformado por herramientas de *software* libre, si bien es cierto, esta opción representa la menor inversión en herramientas para decidirse, sin embargo, es importante, además de su costo, valorar sus limitaciones. Se encuentra que, bajo esta opción , no es posible realizar el *testing* especializado de seguridad, integración y virtualización, ya que solo con herramientas IBM puede llevarse a cabo. Además, el soporte a las mismas no es garantizado, es decir, ante

alguna eventualidad de “caídas” o inconvenientes, se debe esperar a su corrección sin tiempos de espera establecidos, lo que puede comprometer la continuidad de la fábrica.

Se desglosan, a continuación, en la tabla 35, los costos.

Tabla N.º 35

Costos escenario Ecosistema Open

Costo total piloto fábrica de Integration Ecosistema Open	
Licencia Jira	231
Licencia Confluence	25
Licencia Practitest	315
Licencia Jmeter	0
Licencia Katalon	0
Licencia Bitbucket	150
Licencia Github	216
Invision	25
Plant	144
Sketch	48
Total	1154

Nota: elaboración propia.

Costo Ecosistema Mixto Opción 1 y 2.

Tanto la opción 1 como la opción 2 del Ecosistema Mixto proponen una mezcla entre licencias IBM y licencias Open source, la diferencia radica en la herramienta propuesta para la gestión de los defectos, mientras que la opción Mixto 1 propone, para lo que es la práctica de calidad, el paquete completo de licencias IBM; la opción 2 propone, para la gestión de pruebas, un *open source*, lo que implica un proceso manual y tedioso para los *testers* al tener que digitar resultados desde la herramienta IBM al *open source* de gestión.

Tabla N.º 36

Costo escenario Ecosistema Mixto Opción 1

Costo total piloto fábrica de Integration Ecosistema Mixto Opción 1	
Licencia Jira	231
Licencia Confluence	25
Plug-in	572
Licencias testing IBM	37262
Licencia Bitbucket	150
Licencia Github	216
Invision	25
Plant	144
Sketch	48
Total	38673

Nota: elaboración propia.

Tabla N.º 37

Costo escenario Ecosistema Mixto Opción 2

Costo total piloto fábrica de Integration Ecosistema Mixto Opción 2	
Licencia Jira(11 licencias/$7 /usuario/mes)	231
Licencia Confluence	25
Practitest (3 /$35/usuario	315
Licencias testing IBM	36626
Licencia Bitbucket	150
Licencia Github(8/$9/ mes)	216
Invision	25
Plant	144
Sketch	48
Total	37780

Nota: elaboración propia.

Costos totales plan piloto fábrica de Integration

Una vez monetarizados los beneficios y contemplando los costos de mano de obra, materiales y herramientas colaborativas, se obtienen los costos totales de cada escenario, a continuación, se muestra el desglose de los mismos en la tabla 38.

Tabla N.º 38

Costo total piloto fábrica de Integration Ecosistema IBM

Costo total Ecosistema IBM	
Costo mano de obra	468668,81
Costo materiales	134,95
Herramientas colaborativas	44953
Total	558709,76

Nota: elaboración propia.

El costo total del piloto bajo Ecosistema IBM es de $ 558 709,76.

Tabla N.º 39

Costo total piloto fábrica de Integration Ecosistema Open

Costo total piloto fábrica de Integration Ecosistema Open	
Costo Mano de obra	468668,808
Costo materiales	134,95
Costo herramientas colaborativas	1154
Total	471111,758

Nota: elaboración propia.

El costo total del piloto bajo Ecosistema Open es de $ 471 111,758.

Tabla N.º 40

Costo total piloto fábrica de Integration Ecosistema Mixto Opción 1

Costo total piloto fábrica de Integration Ecosistema Mixto Opción 1	
Costo mano de obra	468668,808
Costo materiales	134,95
Costo herramientas colaborativas	38673
Total	546149,758

Nota: elaboración propia.

La opción 1 del Ecosistema Mixto tendría un costo total de $ 546149,758.

Tabla N.º 41

Costo total piloto fábrica de Integration Ecosistema Mixto Opción 2

Costo total piloto fábrica de Integration Ecosistema Mixto Opción 2	
Costo Mano de obra	468668,808
Costo materiales	134,95
Costo herramientas colaborativas	37780
Total	544363,758

Nota: elaboración propia.

La opción 2 del Ecosistema Mixto significaría una inversión de $ 544363,758

Tomando en cuenta las limitaciones que presentan los ecosistemas de *open source*, la necesidad de evitar procesos manuales y la curva de aprendizaje que se debe asumir tanto al inicio del piloto como al replicar el modelo al resto de las células por parte del personal, se recomienda como las mejores opciones por implementar el escenario 1 (Ecosistema IBM), o bien, el escenario 3 (Ecosistema Mixto opción 1), ambos con una relación costo beneficio por encima del 95 %, como se podrá ver a continuación en la tabla 42.

Tabla N.º 42

Relación Costo – Beneficio Escenarios propuestos

Relación Costo Beneficio Escenario 1	96,12%
Relación Costo Beneficio Escenario 2	113,99%
Relación Costo beneficio Escenario 3 Opción 1	98,33%
Relación Costo beneficio Escenario 3 Opción 2	98,65%

Nota: elaboración propia.

Se espera entonces que, al lograr disminuir el tiempo de ciclo del desarrollo de los requerimientos con los mismos recursos que se tienen hoy, sea posible abarcar mayor cantidad de requerimientos, lo que abre posibilidades de poder dar más servicios a los clientes actuales, así

como la posibilidad de capturar nuevos clientes. Igualmente, se alcanzará un ahorro en costos al disminuir las horas del rol del project manager, permitiendo tener más horas disponibles para roles ágiles y operativos que estén totalmente involucrados en la operación.

Posibles limitaciones

-Resistencia al cambio tanto de los recursos como las áreas dentro de GBM de las cuales las fábricas tienen dependencias de servicios.

-Al estar el resto de la organización bajo métodos tradicionales, puede complicarse la operación de la fábrica y toparse con impedimentos que detengan su agilidad.

-Clientes no preparados para el cambio metodológico pueden traer tanto resistencia al cambio como atrasos, mientras se da la madurez necesaria, es requerida la capacitación a nivel de clientes.

-Alcanzar una madurez que refleje los beneficios puede tomar más tiempo, debido a que no solo se está aprendiendo la metodología, sino que se está iniciando la implementación de nuevas herramientas. Se espera que esta última sea simple, debido a las habilidades de los recursos hacia la adopción de medios informáticos.

CONCLUSIONES Y RECOMENDACIONES FINALES

En este proyecto, se presenta una propuesta para el establecimiento de una nueva metodología para desarrollos ágiles en el área de Integration -GBM, la cual va a permitir mejorar la flexibilidad y reducir el *time to market* de los requerimientos. Gracias a ello, las unidades usuarias pueden llegar a contar con desarrollos más oportunos y de mejor calidad, además de lograr una mayor atención de requerimientos.

- El apoyo de la alta gerencia de negocios será un factor clave para que este modelo se institucionalice. La participación permite que el modelo siga mostrando los beneficios obtenidos y evaluar las propuestas de mejoras que se deseen implementar.
- El presupuesto proyectado posibilitará la utilización del modelo Ágil para la atención de los requerimientos, lo cual aminorará los tiempos en la priorización de requerimientos.
- La capacitación en metodología ágil es esencial, ya que permite desarrollar el modelo Ágil de manera eficaz e identifica a los usuarios en la participación activa de los requerimientos. Esta capacitación debe ser constante, no puede ingresar un recurso a las células sin la capacitación necesaria.
- Con la implementación, se espera reducir el riesgo de error y de implementaciones sin valor al negocio.
- La comunicación constante con el *product owner* permite que pueda entender y asimilar los beneficios de este modelo, haciéndolo parte de su día a día. Su participación en el modelo es clave para el éxito de la metodología ágil y con ello se garantiza la calidad del entregable.
- Se propone el uso de indicadores que muestran las mejoras realizadas, además del seguimiento al proceso metodológico.
- Con la propuesta, se han sentado las bases para explorar los usos de metodologías ágiles en el resto de la fábrica de Integration y de la organización de sistemas.

GLOSARIO

Artefacto: nombre dado en prácticas ágiles a las plantillas o elementos colaborativos del proceso.

Auto-organizado: característica de un equipo que lo define como un equipo dirigido y organizado por sus propios miembros, para alcanzar los objetivos especificados por la gerencia, dentro, obviamente, de las limitaciones de su entorno.

Coaching: práctica profesional que consiste en acompañar a un cliente (persona, equipo u organización) a descubrir sus objetivos en un ámbito específico y acompañarle en el descubrimiento propio de la forma de hacerse cargo de ellos y alcanzarlos. La responsabilidad del descubrimiento de los objetivos y la manera de lograrlo es del cliente; el *coach* facilita este descubrimiento.

Consultoría: práctica profesional que consiste en prestar asesoría y consejo profesional a un cliente (persona, equipo u organización), identificando, por una parte, la situación actual y, por otra (de dominio de conocimiento del consultor), la situación deseada; para plantear recomendaciones para el cierre de brechas entre las dos situaciones. El consultor recomienda basado en su conocimiento y experiencia y el cliente aplica las recomendaciones del consultor, quien acompaña al cliente en su aplicación.

Criterios de aceptación: conjunto de criterios, mediante los cuales se valida si una historia de usuario o un elemento de una lista de necesidades fue desarrollada según la expectativa del dueño del producto o del equipo administrador del producto (como representante de los criterios del cliente).

Cross-funcional: característica de un equipo que lo define como un equipo que, uniendo las capacidades y especialidades de cada uno de sus miembros, es capaz de desarrollar completamente una necesidad de negocio.

Defecto: falla en un componente o sistema que podría causar que el componente o sistema no realice su función requerida, por ejemplo: una declaración o definición de datos incorrectos. Un defecto, si es encontrado durante la ejecución, puede causar una interrupción en la ejecución del componente o del sistema.

Definición de listo: también conocida como DoR por su nombre en inglés (*Definition of ready*). Es un conjunto de criterios asociados a una historia de usuario o a un elemento de una lista de necesidades, que deben ser cumplidos antes de iniciar el ciclo de desarrollo y que generan la confianza suficiente a un equipo solucionador, para que este se pueda comprometer a iniciarlo y terminarlo en un período de tiempo determinado.

Definición de terminado: también conocida como DoD por su nombre en inglés (*Definition of done*). Es un conjunto de criterios asociados a una historia de usuario o a un elemento de una lista de necesidades, que deben ser cumplidos al finalizar el ciclo de desarrollo y que generan la confianza suficiente a un equipo solucionador, para que este pueda entregarlo satisfactoriamente al negocio.

Enfoque iterativo e incremental: enfoque de desarrollo de soluciones de *software*, en el que se agrupan necesidades suficientemente valiosas para el negocio, para ser desarrolladas en períodos cortos de tiempo (iteraciones), de tal forma que, en cada iteración, se ejecuta todo el ciclo de desarrollo y se entrega un incremento de producto usable y valioso para el negocio; iteración tras iteración se va incrementando entonces el valor generado para el negocio.

Estimación: en el contexto de las metodologías ágiles, la estimación o estimación relativa es una manera de representar numéricamente y mediante la comparación con un estándar (o pivote), la combinación de esfuerzo, complejidad y riesgo asociado al desarrollo completo de una necesidad representada en una historia de usuario o en un elemento de la lista de necesidades. Esta estimación se suele representar en una escala denominada serie de Fibonacci, utilizando una técnica llamada *planning póker*.

Evento: en el contexto de las metodologías ágiles, un evento es una sesión formal de trabajo colaborativo, realizada periódicamente para lograr un objetivo específico. Los eventos están asociados con marcos de trabajo como Scrum y también suelen ser conocidos como ceremonias.

Facilitación: un proceso por el cual una persona, cuya elección es aceptada por todos los miembros del grupo, que es neutral y no tiene autoridad sustancial en la toma de decisiones, diagnostica e interviene para ayudar al grupo a identificar y resolver problemas y tomar decisiones, para así aumentar su efectividad.

Flujo constante: se refiere al concepto de entregas constantes en el modelo de Scrumban, utilizado para la atención constante de necesidades de mantenimiento y soporte de aplicaciones y que, a diferencia del modelo Scrum, en el que las entregas de incrementos de producto se hacen al finalizar un *sprint*, en este modelo, una vez finalizada la solución de una necesidad, esa es entregada formalmente, una tras otra, de forma constante.

Historia de usuario: es una representación de una necesidad de negocio atómica, escrito de manera concisa, utilizando el lenguaje común del usuario. Las historias de usuario son utilizadas en las metodologías de desarrollo ágiles para la especificación de necesidades (acompañadas siempre de las conversaciones relevantes entre el equipo y los usuarios y de sus respectivos criterios de aceptación).

Inception: conjunto de técnicas y herramientas ágiles utilizadas el iniciar un proyecto, para contextualizar y entender claramente las necesidades y el contexto, así como hacer una delimitación preliminar del alcance y su priorización en función del valor para el negocio.

Iteración: período de tiempo acotado y normalmente fijo, en el que se enmarca el ciclo de desarrollo de necesidades en un marco de trabajo ágil.

Iteración Scrumban: período de tiempo acotado y normalmente fijo, al final del cual se analizan los resultados asociados con el flujo constante de desarrollo de necesidades del equipo en el período.

Kanban: marco de trabajo ágil enfocado en el flujo constante de desarrollo de necesidades, basado en la constante visualización y seguimiento de actividades desde su creación hasta su terminación.

Pivote: punto de partida en el cual se detallan las actividades (desarrollo y pruebas) correspondientes en el desarrollo de la funcionalidad mínima del sistema. Se utiliza como estándar de apoyo para identificar el esfuerzo de los diversos elementos que componen el inventario de necesidades.

Planning poker: técnica colaborativa que se utiliza en las metodologías ágiles para dar una estimación en puntos relativos de las historias de usuario.

Scrum: Scrum es un marco de trabajo para el desarrollo de *software*, en el cual, el desarrollo de la solución a un problema de negocio se lleva a cabo de manera incremental. Se le

da valor al cliente con *software* que funciona en periodos de entrega frecuentes, estableciendo de antemano la periodicidad de las entregas.

Scrumban: la metodología que nace de la combinación de principios de los marcos de trabajo Scrum y Kanban. Es así un marco de trabajo más liviano y enfocado normalmente en proyectos y actividades de mantenimiento y soporte.

Sprint: ciclos de iteración en la construcción de *software* enmarcado en marcos de trabajos ágiles como Scrum.

REFERENCIAS

Acuña, J. (2012). *Control de Calidad: Un enfoque integral y estadístico.* Cartago: Tecnológica de Costa Rica.

Arias, F. (2012). *El Proyecto de Investigación: Introducción a la metodología científica. Sexta edición.* Episteme.

Atlassian. (2017). *Atlassian JIRA Software*. Recuperado de https://es.atlassian.com

Beck, K. (2001). *Ágile manifiesto.* Recuperado de http://agilemanifesto.org/

Beck, K., & Andrés, C. (2005). *Extreme Programming Explained: Embraice Change.* US: Pearson Education, INC.

Bitbucket. (2017). *Bitbucket.org*. Recuperado de https://bitbucket.org/

Caso de éxito Banca Peruana. (2017).

CHAOS. (2015). *CHAOS Report, CHAOS database.* Standish Group.

Cohen, J., Teleki, S., & Brown, E. (2013). *Best kept secrets peer code review.* US: SmartBear Software.

Cruelles, J. (2013). *Ingeniería Industrial: Métodos de trabajo tiempos, y su aplicación a la planificación y a la mejora continua.* México: Alfaomega, Marcombo.

Cuevas, A., Méndez, S y Hernández, R. (2014). *Introducción al estilo APA para citas y referencias.* Tercera edición. México: Universidad de Celaya. Recuperado de http://www.udec.edu.mx/i2012/investigacion/investigacion.html

David, F. (2008). *Conceptos de Administración Estratégica.* Décimo primera edición. México: Pearson.

GitHub. (2017). *GitHub is how people build software*. Recuperado de https://github.com/github

Heizer, J., & Render, B. (2009). *Administración de Operaciones.* Sétima edición. México: Pearson -Prentice Hall.

Hernández, R. (2017). *Fundamentos de investigación.* México: Mc Graw Hill.

Hill, C., & Jones, G. (2009). *Administración Estratégica.* Octava Edición. Mc Graw Hill.

Hundermark, P. (2015). *Do Better Scrum.* US: InfoQ.

IBM. (2016). *Design Thinking Field Guide*. Recuperado de https://ibm.ent.box.com/s/dw2j8nnmj99446my8vgw26subrs0ztg9/file/92578560113

IBM. (2017a). *Blue Mix Devops*. Recuperado de https://console.bluemix.net/devops

IBM. (2017b). *Design Thinking*. Recuperado de https://www.ibm.com/design/thinking/

IBM. (2017c). *Haga de los procesos algo sencillo*. Recuperado de http://www-01.ibm.com/software/mx/info/clients/blueworks/bwlcampaign.html

Index, BTM Business Agility. (2010). *BTM Research Report: The Characteristics of an Agile Enterprise and How They Drive Superior Financial Performance by Converging Business and Technology Management.*

Jeffries, R. (2017). *Extreme Programming.* Recuperado de https://ronjeffries.com/search.html

Kanawaty, G. (1996). *Introducción al Estudio del Trabajo.* Ginebra: OIT.

Kniberg, H. (2007). *Scrum and XP from the Trenches.* Info Q.

Lamb, Hair, & McDaniel. (2011). *Marketing.* Mason. OH USA: South Western Cengage Learning.

Microsoft. (2017a). *Microsoft*. Recuperado de https://www.microsoft.com/es-cr/

Microsoft. (2017b). *Products.office: Visio online*. Recuperado de https://products.office.com/es/visio/visio-online

Montvelisky, J. (2017). *Practitest*. Recuperado de https://www.practitest.com/

Osherove, R. (2014). *The arte of unit testing.* Segunda edición. Manning Shelter Island.

Palacios, L. (2009). *Ingeniería de métodos movimientos y tiempos.* Bogotá: Ecoe Ediciones.

Pichler, R. (2012). *Agile product management with scrum: creating products that customers love.* Boston,US: Addison-Wesley, Pearson Education.

PMI. (2017). *Pulse of the profession.*

PMP. (s.f.). *Pulse of the* .

Pressman, R. (2010). *Ingeniería del Software: Un enfoque práctico.* México: Mc Graw Hill.

Rasmusson, J. (2010). *The Agile Samurai.* Dallas, US: Pragmatic Bookshelf.

Rubin, K. (2012). *Essencial Scrum: A practical Guide to the most popular agile process.* Michigan, US: Addison-Wesley, Pearson Education.

Satpathy, T. (2016). *Una Guía para el cuerpo de conocimiento Scrum, Guía Sbook.* Recuperado de https://www.scrumstudy.com/SBOK/SCRUMstudy-SBOK-Guide-2016-spanish.pdf

Schwaber & Sutherland. (2017). *La Guía Definitiva de Scrum: Las reglas del juego.* Recuperado de http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100

Shore, J., & Warden, S. (2008). *The art of agile development.* California, US: O'Reilly Media, Inc.

Swisher, W. (2014). *Implementing Scrumban, A short guide to implementing Scrumban at your organization.* US.

Wells, D. (2013). *Extreme programming.* Recuperado de http://www.extremeprogramming.org/more.html

Zephyr. (2016). *Zephyr Test Management*. Recuperado de https://www.getzephyr.com/

APÉNDICES

Apéndice No 1. Guía para conversatorio *personal Integration*

Familia	Consulta	Comentarios
Organización	¿Cuál es la estructura de su práctica?	
	Descripción de roles	
	¿Dónde inicia y dónde termina el proceso?	
	¿Cómo están organizados para atender las solicitudes?	
	¿Cómo les solicitan los servicios?	
	¿Qué tipo de solicitudes trabajan?	
Operación	¿Qué productos comercializan?	
	¿Cómo se categorizan los expertos por producto?	
	¿Quiénes integran los equipos de trabajo, quiénes y cómo lideran?	
	¿Cómo asignan a los recursos?	
	¿Cuáles *tools* de gestión utilizan?	
	¿Cómo se planifica la producción?	
	¿Cuáles son los insumos de producción?	
	¿Cómo se miden los procesos? (Métricas)	
	Artefactos	
	Ceremonias	
	Entregables	
	¿Cómo se interactúa con otras áreas (PMO, Arquitectura, Calidad y otras prácticas/temáticas)?	
	¿Cuál es la cantidad de RQ desarrollados al mes?	
	¿Se conoce si existen desperdicios (retrabajo y tiempo muertos)?	
	Cantidad de defectos	
Mejora continua	¿Qué aspectos se están haciendo bien?	
	¿Qué aspectos se tienen que mejorar y por qué?	
	¿Qué se tiene que mantener?	
	¿Cuál cree que debe ser el proceso de la fábrica?	

Si la población es finita, es decir, se conoce el total de la población y se deseara saber cuántos del total se tendrá que estudiar, la fórmula sería:

Donde:

• N = Total de la población

• Zα= 1.96 al cuadrado (si la seguridad es del 95%)

• p = proporción esperada (0.05)

• q = 1 – p (en este caso 1-0.05 = 0.95)

• d = precisión (5%).

$$n = \frac{N * Z_{\alpha}^{2} p * q}{d^{2} * (N-1) + Z_{\alpha}^{2} * p * q}$$

Cálculo de Muestras para Poblaciones Finitas

INGRESO DE PARAMETROS	
Tamaño de la Población (N)	40
Error Muestral (E)	0,05
Proporción de Éxito (P)	0,05
Proporción de Fracaso (Q)	0,95
Valor para Confianza (Z)	1,96

	Tamaño de Muestra
Muestra Optima	26

Apéndice No 2. Instrumentos para evaluar el nivel de madurez de las prácticas de desarrollo de *software*

Evaluación Madurez Práctica Levantamiento Requerimientos

Criterio	#	Pregunta	Equipo SO										
			Total Si	Total No	Total N/A	Total	% Si	% No	%N/A	% Si	% No	% N/A	% Total
Nivel de madurez del equipo en aspectos de levantamiento de requerimientos	1	Hay una etapa definida para el levantamiento de requerimientos	0	0	0	0	0%	0%	0%	0%	0%	0%	0%
	2	Si es producto nuevo, se analiza el problema que se tenga entes de ver la solución	0	0	0	0	0%	0%	0%				
	3	En el levantamiento de Requerimientos, definen el Producto Minimo Viable	0	0	0	0	0%	0%	0%				
	4	Quien levanta requerimientos no desarrolla	0	0	0	0	0%	0%	0%				
	5	Se usa un mismo formato para el levantamiento de requerimientos en todos los proyectos	0	0	0	0	0%	0%	0%				
	6	Poseen un método para documentar dependencias entre los requerimientos	0	0	0	0	0%	0%	0%				
Nivel de claridad de los requerimientos proporcionados a los desarrolladores	7	Los requisitos son analizados por el Team Lead, con el objetivo de velar por los posibles riesgos o impedimentos relacionados a los RQs	0	0	0	0	0%	0%	0%	0%	0%	0%	0%
	8	Los desarrolladores cuentan con toda la información necesaria para desarrollar	0	0	0	0	0%	0%	0%				
	9	El equipo entiende bien todos los requerimientos que son proporcionados	0	0	0	0	0%	0%	0%				
Nivel de conocimiento y práctica del uso de las metodologías ágiles en el levantamiento de requerimientos.	10	Conoce que es una historia de usuario	0	0	0	0	0%	0%	0%	0%	0%	0%	0%
	11	Los requerimientos son expresados en historias de usuario	0	0	0	0	0%	0%	0%				
	12	Las historias de usuario poseen respectivos criterios de aceptación	0	0	0	0	0%	0%	0%				
	13	Las historias de usuario o bien los requerimientos poseen ID	0	0	0	0	0%	0%	0%				

Evaluación Madurez Práctica Aseguramiento y Control de Calidad

	#	Pregunta	Total SI	Total No	Total	% Si	% No	Por Criterio % Si	Por Criterio % No
Nivel de involucramiento temprano de calidad en el proceso	1	Existe un involucramiento temprano de los QA Enginners en sus proyectos	0	0	0	0%	0%	0	0
	2	Los requisitos son analizados por el QA Engineer, con el objetivo de velar por los posibles riesgos, impedimentos, necesidades especiales, y/o identificación de datos de para pruebas	0	0	0	0%	0%		
	3	Los QA Engineers cuentan con toda la información necesaria para desarrollar sus pruebas	0	0	0	0%	0%		
	4	Se identifica en etapas tempranas la necesidad de pruebas no funcionales (rendimiento, recuperación, etc)	0	0	0	0%	0%		
Nivel de madurez del dpto en las prácticas de calidad	5	Existe un proceso definido y estándar de desarrollo en su unidad	0	0	0	0%	0%	0	0
	6	Existe un proceso definido y estándar de calidad en su unidad	0	0	0	0%	0%		
	7	Existe un proceso definido de pruebas en sus equipos	0	0	0	0%	0%		
	8	Se diseñan pruebas unitarias	0	0	0	0%	0%		
	9	Se ejecutan pruebas unitarias	0	0	0	0%	0%		
	10	Se documentan los resultados de las pruebas unitarias	0	0	0	0%	0%		
	11	Se cuenta con criterios de aceptación para cada HU o RQ	0	0	0	0%	0%		
	12	Son los criterios de aceptación 100% claros, sin ambiguedades y alineados a asegurar el cumplimiento de la necesidad del negocio	0	0	0	0%	0%		
	13	Se documentan los cambios realizados por el cliente una vez iniciado el proyecto	0	0	0	0%	0%		
	14	Se especifica el impacto de los cambios solicitados por el cliente	0	0	0	0%	0%		
	15	Se diseña un plan de pruebas para cada proyecto	0	0	0	0%	0%		
	16	Se diseñan casos de pruebas en cada proyecto	0	0	0	0%	0%		
	17	Existe un estándar de nomenclatura para los objetos creados	0	0	0	0%	0%	0	0
Nivel de madurez del cierre del proceso de calidad y el uso de indicadores	18	Se documentan e informan los defectos utilizando una herramienta de seguimiento de defectos y/o reportes en documentos de office.	0	0	0	0%	0%		
	19	Se monitorean la resolución de defectos y los esfuerzos necesarios para ello	0	0	0	0%	0%		
	20	Se realizan pruebas no funcionales en sus desarrollos de forma proactiva	0	0	0	0%	0%		
	21	Se realizan pruebas no funcionales en sus desarrollos de forma reactiva	0	0	0	0%	0%		
	22	Se documentan e informan los resultados de las pruebas no funcionales realizadas	0	0	0	0%	0%		
	23	Se documenta un informe final de calidad	0	0	0	0%	0%		
	24	Se utilizan kpi's en el equipo con el fin de contar con información para mejora contínua	0	0	0	0%	0%		

Evaluación Madurez Práctica de Release Management

Criterios	#	Pregunta	IC									
			Total Si	Total No	Total	% Si	% No	%N/A	% Si	% No	%N/A	% Total
Nivel de conocimiento sobre prácticas de Release Management	1	Cuentan con conocimientos básicos de gestión de versiones	0	0	0	0%	0%	0%	0%	0%	0%	0%
	2	Conocen los elementos fundamentales de un repositorio	0	0	0	0%	0%	0%				
	3	Cuentan con conocimiento en el software GIT	0	0	0	0%	0%	0%				
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	4	Cuentan con proyectos en repositorios	0	0	0	0%	0%	0%	0%	0%	0%	0%
	5	Cuentan con una herramienta para realizar los release de sus proyectos	0	0	0	0%	0%	0%				
	6	Cuentan con estándares de nomenclatura para sus objetos	0	0	0	0%	0%	0%				
	7	Existe un estándar para la creación de repositorios	0	0	0	0%	0%	0%				
	8	Existe un control de notificaciones hacia el cliente cada vez que realizan un release	0	0	0	0%	0%	0%				
	9	Existe algun incoveniente para utilizar la metodología de release	0	0	0	0%	0%	0%				
Nivel de planificación actual para el versionamiento	10	Tienen una hora especifica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio	0	0	0	0%	0%	0%	0%	0%	0%	0%
	11	Planifican los release al iniciar el proyecto	0	0	0	0%	0%	0%				

Apéndice No 3. Resultados de la evaluación de madurez en prácticas de desarrollo por equipo y práctica

Evaluación de RQs - IC

Objetivo	#	Pregunta	Integración										
			Total Si	Total No	Total N/A	Total	% Si	% No	%N/A	% Si	% No	% N/A	% Total
Analizar la madurez del equipo en aspectos de levantamiento de requerimientos	1	Hay una etapa definida para el levantamiento de requerimientos	11	0	0	11	100%	0%	0%	45%	55%	0%	100%
	2	Si es producto nuevo, se analiza el problema que se tenga entes de ver la solución	9	2	0	11	82%	18%	0%				
	3	En el levantamiento de Requerimientos, definen el Producto Minimo Viable	0	11	0	11	0%	100%	0%				
	4	Quien levanta requerimientos no desarrolla	7	4	0	11	64%	36%	0%				
	5	Se usa un mismo formato para el levantamiento de requerimientos en todos los proyectos	3	8	0	11	27%	73%	0%				
	6	Poseen un método para documentar dependencias entre los requerimientos	0	11	0	11	0%	100%	0%				
Evaluar los requerimientos proporcionados a los desarrolladores	7	Los requisitos son analizados por el Team Lead, con el objetivo de velar por los posibles riesgos o impedimentos relacionados a los RQs	5	6	0	11	45%	55%	0%	24%	76%	0%	100%
	8	Los desarrolladores cuentan con toda la información necesaria para desarrollar	2	9	0	11	18%	82%	0%				
	9	El equipo entiende bien todos los requerimientos que son proporcionados	1	10	0	11	9%	91%	0%				
Evaluar el conocimiento y la práctica del uso de las metodologías ágiles en el levantamiento de requerimientos.	10	Conoce que es una historia de usuario	5	6	0	11	45%	55%	0%	27%	73%	0%	100%
	11	Los requerimientos son expresados en historias de usuario	0	11	0	11	0%	100%	0%				
	12	Las historias de usuario o requerimientos poseen respectivos criterios de aceptación	4	7	0	11	36%	64%	0%				
	13	Las historias de usuario o bien los requerimientos poseen ID	3	8	0	11	27%	73%	0%				

Evaluación de RQs - BA

Objetivo	#	Pregunta	IT Platform										
			Total Si	Total No	Total N/A	Total	% Si	% No	%N/A	% Si	% No	% N/A	% Total
Analizar la madurez del equipo en aspectos de levantamiento de requerimientos	1	Hay una etapa definida para el levantamiento de requerimientos	14	3	0	17	82%	18%	0%	43%	57%	0%	100%
	2	Si es producto nuevo, se analiza el problema que se tenga antes de ver la solución	10	7	0	17	59%	41%	0%				
	3	En el levantamiento de Requerimientos, definen el Producto Minimo Viable	0	17	0	17	0%	100%	0%				
	4	Quien levanta requerimientos no desarrolla	0	17	0	17	0%	100%	0%				
	5	Se usa un mismo formato para el levantamiento de requerimientos en todos los proyectos	9	8	0	17	53%	47%	0%				
	6	Poseen un método para documentar dependencias entre los requerimientos	11	6	0	17	65%	35%	0%				
Evaluar los requerimientos proporcionados a los desarrolladores	7	Los requisitos son analizados por el Team Lead, con el objetivo de velar por los posibles riesgos o impedimentos relacionados a los RQs	0	17	0	17	0%	100%	0%	24%	76%	0%	100%
	8	Los desarrolladores cuentan con toda la información necesaria para desarrollar	7	10	0	17	41%	59%	0%				
	9	El equipo entiende bien todos los requerimientos que son proporcionados	5	12	0	17	29%	71%	0%				
Evaluar el conocimiento y la práctica del uso de las metodologías ágiles en el levantamiento de requerimientos.	10	Conoce que es una historia de usuario	12	5	0	17	71%	29%	0%	32%	68%	0%	100%
	11	Los requerimientos son expresados en historias de usuario	0	17	0	17	0%	100%	0%				
	12	- Las historias de usuario poseen respectivos criterios de aceptación	0	17	0	17	0%	100%	0%				
	13	- Las historias de usuario o bien los requerimientos poseen ID	10	7	0	17	59%	41%	0%				

Evaluación de RQs - Equipo SO

Objetivo	#	Pregunta	SO										
			Total Si	Total No	Total N/A	Total	% Si	% No	%N/A	% Si	% No	% N/A	% Total
Analizar la madurez del equipo en aspectos de levantamiento de requerimientos	1	Hay una etapa definida para el levantamiento de requerimientos	12	0	0	12	100%	0%	0%	47%	53%	0%	100%
	2	Si es producto nuevo, se analiza el problema que se tenga entes de ver la solución	8	4	0	12	67%	33%	0%				
	3	En el levantamiento de Requerimientos, definen el Producto Minimo Viable	0	12	0	12	0%	100%	0%				
	4	Quien levanta requerimientos no desarrolla	2	10	0	12	17%	83%	0%				
	5	Se usa un mismo formato para el levantamiento de requerimientos en todos los proyectos	9	3	0	12	75%	25%	0%				
	6	Poseen un método para documentar dependencias entre los requerimientos	3	9	0	12	25%	75%	0%				
Evaluar los requerimientos proporcionados a los desarrolladores	7	Los requisitos son analizados por el Team Lead, con el objetivo de velar por los posibles riesgos o impedimentos relacionados a los RQs	9	3	0	12	75%	25%	0%	44%	56%	0%	100%
	8	Los desarrolladores cuentan con toda la información necesaria para desarrollar	4	8	0	12	33%	67%	0%				
	9	El equipo entiende bien todos los requerimientos que son proporcionados	3	9	0	12	25%	75%	0%				
Evaluar el conocimiento y la práctica del uso de las metodologías ágiles en el levantamiento de requerimientos.	10	Conoce que es una historia de usuario	6	6	0	12	50%	50%	0%	40%	60%	0%	100%
	11	Los requerimientos son expresados en historias de usuario	0	12	0	12	0%	100%	0%				
	12	- Las historias de usuario o requerimientos poseen respectivos criterios de aceptación	5	7	0	12	42%	58%	0%				
	13	Las historias de usuario o bien los requerimientos poseen ID	8	4	0	12	67%	33%	0%				

Evaluación Aseguramiento y Control de Calidad Equipo IC

	#	Pregunta	IC					Por Criterio	
			Total Si	Total No	Total	% Si	% No	% Si	% No
Nivel de involucramiento temprano de calidad en el proceso	1	Existe un involucramiento temprano de los QA Enginners en sus proyectos	7	4	11	64%	36%	36	64
	2	Los requisitos son analizados por el QA Engineer, con el objetivo de velar por los posibles riesgos, impedimentos, necesidades especiales, y/o identificación de datos de para pruebas	4	7	11	36%	64%		
	3	Los QA Engineers cuentan con toda la información necesaria para desarrollar sus pruebas	2	9	11	18%	82%		
	4	Se identifica en etapas tempranas la necesidad de pruebas no funcionales (rendimiento, recuperación, etc)	3	8	11	27%	73%		
Nivel de madurez del dpto en las prácticas de calidad	5	Existe un proceso definido y estándar de desarrollo en su unidad	10	1	11	91%	9%	48	52
	6	Existe un proceso definido y estándar de calidad en su unidad	8	3	11	73%	27%		
	7	Existe un proceso definido de pruebas en sus equipos	0	11	11	0%	100%		
	8	Se diseñan pruebas unitarias	4	7	11	36%	64%		
	9	Se ejecutan pruebas unitarias	4	7	11	36%	64%		
	10	Se documentan los resultados de las pruebas unitarias	4	7	11	36%	64%		
	11	Se cuenta con criterios de aceptación para cada HU o RQ	4	7	11	36%	64%		
	12	Son los criterios de aceptación 100% claros, sin ambiguedades y alineados a asegurar el cumplimiento de la necesidad del negocio	9	2	11	82%	18%		
	13	Se documentan los cambios realizados por el cliente una vez iniciado el proyecto	5	6	11	45%	55%		
	14	Se especifica el impacto de los cambios solicitados por el cliente	8	3	11	73%	27%		
	15	Se diseña un plan de pruebas para cada proyecto	5	6	11	45%	55%		
	16	Se diseñan casos de pruebas en cada proyecto	3	8	11	27%	73%		
	17	Existe un estándar de nomenclatura para los objetos creados	0	11	11	0%	100%		
Nivel de madurez del cierre del proceso de calidad y el uso de indicadores	18	Se documentan e informan los defectos utilizando una herramienta de seguimiento de defectos y/o reportes en documentos de office.	4	7	11	36%	64%	35	65
	19	Se monitorean la resolución de defectos y los esfuerzos necesarios para ello	5	6	11	45%	55%		
	20	Se realizan pruebas no funcionales en sus desarrollos de forma proactiva	3	8	11	27%	73%		
	21	Se realizan pruebas no funcionales en sus desarrollos de forma reactiva	7	4	11	64%	36%		
	22	Se documentan e informan los resultados de las pruebas no funcionales realizadas	7	4	11	64%	36%		
	23	Se documenta un informe final de calidad	5	6	11	45%	55%		
	24	Se utilizan kpi's en el equipo con el fin de contar con información para mejora contínua	0	11	11	0%	100%		

Evaluación de Aseguramiento y Control de Calidad Equipo BA

Criterios	#	Pregunta	BA					Por objetivo	
			Total Si	Total No	Total	% Si	% No	% Si	% No
Nivel de involucramiento temprano de calidad en el proceso	1	Existe un involucramiento temprano de los QA Enginners en sus proyectos	8	4	12	67%	33%	38	63
	2	Los requisitos son analizados por el QA Engineer, con el objetivo de velar por los posibles riesgos, impedimentos, necesidades especiales, y/o identificación de datos de para pruebas	3	9	12	25%	75%		
	3	Los QA Engineers cuentan con toda la información necesaria para desarrollar sus pruebas	5	7	12	42%	58%		
	4	Se identifica en etapas tempranas la necesidad de pruebas no funcionales (rendimiento, recuperación, etc)	2	10	12	17%	83%		
Nivel de madurez del dpto en las prácticas de calidad	5	Existe un proceso definido y estándar de desarrollo en su unidad	9	3	12	75%	25%	39	61
	6	Existe un proceso definido y estándar de calidad en su unidad	6	6	12	50%	50%		
	7	Existe un proceso definido de pruebas en sus equipos	6	6	12	50%	50%		
	8	Se diseñan pruebas unitarias	5	7	12	42%	58%		
	9	Se ejecutan pruebas unitarias	3	9	12	25%	75%		
	10	Se documentan los resultados de las pruebas unitarias	3	9	12	25%	75%		
	11	Se cuenta con criterios de aceptación para cada HU o RQ	0	12	12	0%	100%		
	12	Son los criterios de aceptación 100% claros, sin ambiguedades y alineados a asegurar el cumplimiento de la necesidad del negocio	0	12	12	0%	100%		
	13	Se documentan los cambios realizados por el cliente una vez iniciado el proyecto	8	4	12	67%	33%		
	14	Se especifica el impacto de los cambios solicitados por el cliente	6	6	12	50%	50%		
	15	Se diseña un plan de pruebas para cada proyecto	5	7	12	42%	58%		
	16	Se diseñan casos de pruebas en cada proyecto	5	7	12	42%	58%		
Nivel de madurez del cierre del proceso de calidad y el uso de indicadores	17	Existe un estándar de nomenclatura para los objetos creados	6	6	12	50%	50%	44	56
	18	Se documentan e informan los defectos utilizando una herramienta de seguimiento de defectos y/o reportes en documentos de office.	5	7	12	42%	58%		
	19	Se monitorean la resolución de defectos y los esfuerzos necesarios para ello	4	8	12	33%	67%		
	20	Se realizan pruebas no funcionales en sus desarrollos de forma proactiva	4	8	12	33%	67%		
	21	Se realizan pruebas no funcionales en sus desarrollos de forma reactiva	10	2	12	83%	17%		
	22	Se documentan e informan los resultados de las pruebas no funcionales realizadas	8	4	12	67%	33%		
	23	Se documenta un informe final de calidad	4	8	12	33%	67%		
	24	Se utilizan kpi's en el equipo con el fin de contar con información para mejora contínua	1	11	12	8%	92%		

Evaluación Aseguramiento y Control de Calidad Equipo SO

	#	Pregunta	SO					Por Criterio	
			Total Si	Total No	Total	% Si	% No	% Si	% No
Nivel de involucramiento temprano de calidad en el proceso	1	Existe un involucramiento temprano de los QA Enginners en sus proyectos	5	7	12	42%	58%	29	71
	2	Los requisitos son analizados por el QA Engineer, con el objetivo de velar por los	4	8	12	33%	67%		
	3	Los QA Engineers cuentan con toda la información necesaria para desarrollar sus pruebas	3	9	12	25%	75%		
	4	Se identifica en etapas tempranas la necesidad de pruebas no funcionales	2	10	12	17%	83%		
Nivel de madurez del dpto en las prácticas de calidad	5	Existe un proceso definido y estándar de desarrollo en su unidad	8	4	12	67%	33%	40	60
	6	Existe un proceso definido y estándar de calidad en su unidad	5	7	12	42%	58%		
	7	Existe un proceso definido de pruebas en sus equipos	5	7	12	42%	58%		
	8	Se diseñan pruebas unitarias	4	8	12	33%	67%		
	9	Se ejecutan pruebas unitarias	2	10	12	17%	83%		
	10	Se documentan los resultados de las pruebas unitarias	3	9	12	25%	75%		
	11	Se cuenta con criterios de aceptación para cada HU o RQ	5	7	12	42%	58%		
	12	Son los criterios de aceptación 100% claros, sin ambiguedades y alineados a asegurar el cumplimiento de la necesidad del negocio	3	9	12	25%	75%		
	13	Se documentan los cambios realizados por el cliente una vez iniciado el proyecto	8	4	12	67%	33%		
	14	Se especifica el impacto de los cambios solicitados por el cliente	8	4	12	67%	33%		
	15	Se diseña un plan de pruebas para cada proyecto	3	9	12	25%	75%		
	16	Se diseñan casos de pruebas en cada proyecto	3	9	12	25%	75%		
Nivel de madurez del cierre del proceso de calidad y el uso de indicadores	17	Existe un estándar de nomenclatura para los objetos creados	2	10	12	17%	83%	19	81
	18	Se documentan e informan los defectos utilizando una herramienta de	0	12	12	0%	100%		
	19	Se monitorean la resolución de defectos y los esfuerzos necesarios para ello	0	12	12	0%	100%		
	20	Se realizan pruebas no funcionales en sus desarrollos de forma proactiva	5	7	12	42%	58%		
	21	Se realizan pruebas no funcionales en sus desarrollos de forma reactiva	8	4	12	67%	33%		
	22	Se documentan e informan los resultados de las pruebas no funcionales realizadas	3	9	12	25%	75%		
	23	Se documenta un informe final de calidad	0	12	12	0%	100%		
	24	Se utilizan kpi's en el equipo con el fin de contar con información para mejora contínua	0	12	12	0%	100%		

Evaluación de Release Management - IC

Criterios	#	Pregunta	IC									
			Total Si	Total No	Total	% Si	% No	%N/A	% Si	% No	%N/A	% Total
Nivel de conocimiento sobre prácticas de Release Management	1	Cuentan con conocimientos básicos de gestión de versiones	9	2	11	82%	18%	0%	67%	33%	0%	100%
	2	Conocen los elementos fundamentales de un repositorio	9	2	11	82%	18%	0%				
	3	Cuentan con conocimiento en el software GIT	4	7	11	36%	64%	0%				
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	4	Cuentan con proyectos en repositorios	7	4	11	64%	36%	0%	45%	55%	0%	100%
	5	Cuentan con una herramienta para realizar los release de sus proyectos	5	6	11	45%	55%	0%				
	6	Cuentan con estándares de nomenclatura para sus objetos	6	5	11	55%	45%	0%				
	7	Existe un estándar para la creación de repositorios	4	7	11	36%	64%	0%				
	8	Existe un control de notificaciones hacia el cliente cada vez que realizan un release	5	6	11	45%	55%	0%				
	9	Existe algun incoveniente para utilizar la metodología de release	3	8	11	27%	73%	0%				
Nivel de planificación actual para el versionamiento	10	Tienen una hora especifica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio	4	7	11	36%	64%	0%	32%	68%	0%	100%
	11	Planifican los release al iniciar el proyecto	3	8	11	27%	73%	0%				

Evaluación de Release Management - BA

Criterios	#	Pregunta	BA							
			Total Si	Total No	Total	% Si	% No	% Si	% No	% Total
Nivel de conocimiento sobre prácticas de Release Management	1	Cuentan con conocimientos básicos de gestión de versiones	10	7	17	59%	41%	55%	45%	100%
	2	Conocen los elementos fundamentales de un repositorio	9	8	17	53%	47%			
	3	Cuentan con conocimiento en el software GIT	9	8	17	53%	47%			
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	4	Cuentan con proyectos en repositorios	8	9	17	47%	53%	40%	60%	100%
	5	Cuentan con una herramienta para realizar los release de sus proyectos	6	11	17	35%	65%			
	6	Cuentan con estándares de nomenclatura para sus objetos	6	11	17	35%	65%			
	7	Existe un estándar para la creación de repositorios	8	9	17	47%	53%			
	8	Existe un control de notificaciones hacia el cliente cada vez que realizan un release	10	7	17	59%	41%			
	9	Existe algun incoveniente para utilizar la metodología de release	3	14	17	18%	82%			
Nivel de planificación actual para el versionamiento	10	Tienen una hora especifica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio	10	7	17	59%	41%	50%	50%	100%
	11	Planifican los release al iniciar el proyecto	7	10	17	41%	59%			

Evaluación de Release Management - Equipo SO

Criterios	#	Pregunta	Total Si	Total No	Total	% Si	% No	% Si	% No	% Total
Nivel de conocimiento sobre prácticas de Release Management	1	Cuentan con conocimientos básicos de gestión de versiones	10	2	12	83%	17%	81%	19%	100%
	2	Conocen los elementos fundamentales de un repositorio	10	2	12	83%	17%			
	3	Cuentan con conocimiento en el software GIT	9	3	12	75%	25%			
Nivel de estandarización de metodologías establecidas sobre Release Management en el equipo	4	Cuentan con proyectos en repositorios	8	4	12	67%	33%	51%	49%	100%
	5	Cuentan con una herramienta para realizar los relase de sus proyectos	5	7	12	42%	58%			
	6	Cuentan con estándares de nomenclatura para sus objetos	6	6	12	50%	50%			
	7	Existe un estándar para la creación de repositorios	8	4	12	67%	33%			
	8	Existe un control de notificaciones hacia el cliente cada vez que realizan un release	8	4	12	67%	33%			
	9	Existe algun incoveniente para utilizar la metodología de release	2	10	12	17%	83%			
Nivel de planificación actual para el versionamiento	10	Tienen una hora especifica para subir el desarrollo de cambios o de nuevas funcionalidades al repositorio	5	7	12	42%	58%	33%	67%	100%
	11	Planifican los release al iniciar el proyecto	3	9	12	25%	75%			

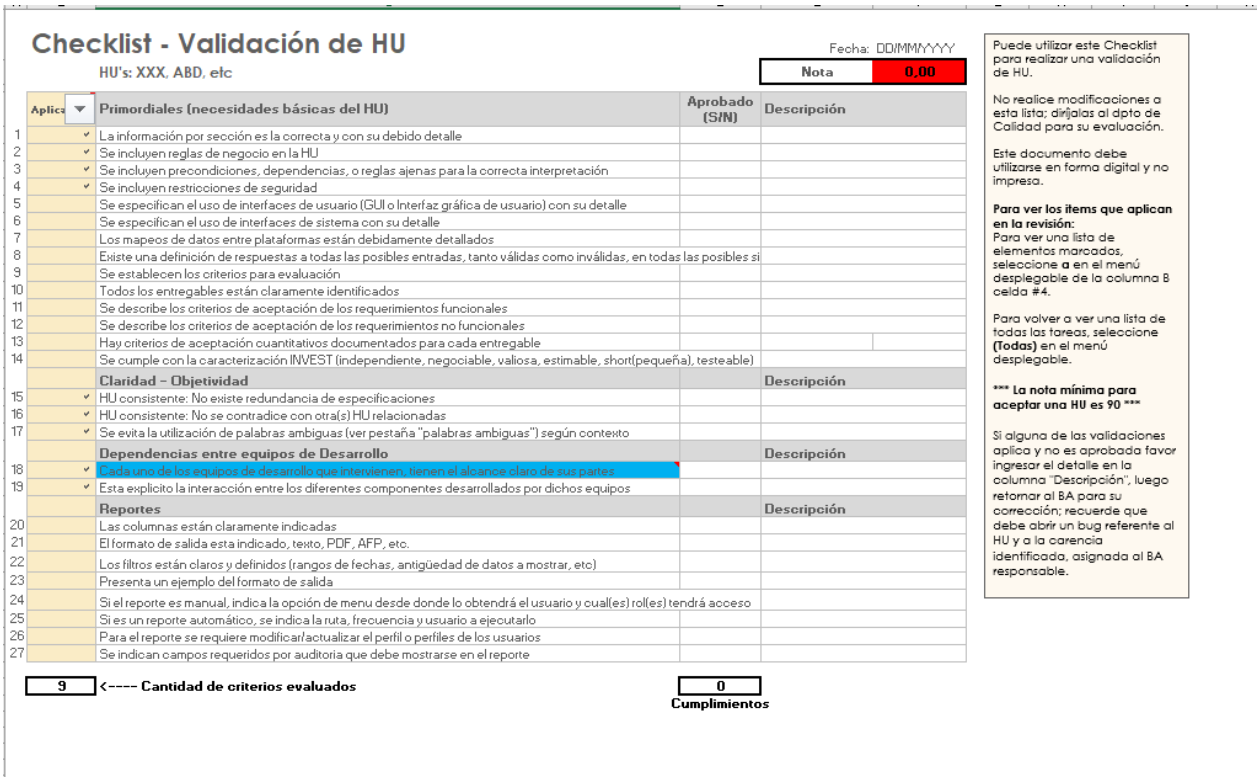
Apéndice No 4. Historia de usuario

[Línea de producto] - [Modulo]

[Nombre Historia de Usuario]

Tipo	Mantenimiento ☐	Nueva Funcionalidad ☐	
ID Historia de Usuario	HU- XX	**HU Relacionada**	HU- XX
Prioridad de negocio	Alta ☐	Media ☐	Baja ☐
Yo como	<Rol>		
Deseo	<Necesidad>		
Para	<Valor que le genera al negocio>		
Criterios de Aceptación	<Expectativas que espera el dueño del producto>		
Elementos de conversación	<Conclusiones que se generan en la conversaciones entre área>		
Anexos	<Todo aquello que sea de importancia para dar valor al desarrollo>		

Apéndice No 5. *Check list* de Calidad



Checklist - Validación de HU

HU's: XXX, ABD, etc

Fecha: DD/MM/YYYY

Nota 0,00

	Aplica	Primordiales (necesidades básicas del HU)	Aprobado (S/N)	Descripción
1		La información por sección es la correcta y con su debido detalle		
2		Se incluyen reglas de negocio en la HU		
3		Se incluyen precondiciones, dependencias, o reglas ajenas para la correcta interpretación		
4		Se incluyen restricciones de seguridad		
5		Se especifican el uso de interfaces de usuario (GUI o Interfaz gráfica de usuario) con su detalle		
6		Se especifican el uso de interfaces de sistema con su detalle		
7		Los mapeos de datos entre plataformas están debidamente detallados		
8		Existe una definición de respuestas a todas las posibles entradas, tanto válidas como inválidas, en todas las posibles si		
9		Se establecen los criterios para evaluación		
10		Todos los entregables están claramente identificados		
11		Se describe los criterios de aceptación de los requerimientos funcionales		
12		Se describe los criterios de aceptación de los requerimientos no funcionales		
13		Hay criterios de aceptación cuantitativos documentados para cada entregable		
14		Se cumple con la caracterización INVEST (independiente, negociable, valiosa, estimable, short(pequeña), testeable)		
		Claridad – Objetividad		**Descripción**
15		HU consistente: No existe redundancia de especificaciones		
16		HU consistente: No se contradice con otra(s) HU relacionadas		
17		Se evita la utilización de palabras ambiguas (ver pestaña "palabras ambiguas") según contexto		
		Dependencias entre equipos de Desarrollo		**Descripción**
18		Cada uno de los equipos de desarrollo que intervienen, tienen el alcance claro de sus partes		
19		Esta explicito la interacción entre los diferentes componentes desarrollados por dichos equipos		
		Reportes		**Descripción**
20		Las columnas están claramente indicadas		
21		El formato de salida esta indicado, texto, PDF, AFP, etc.		
22		Los filtros están claros y definidos (rangos de fechas, antigüedad de datos a mostrar, etc)		
23		Presenta un ejemplo del formato de salida		
24		Si el reporte es manual, indica la opción de menu desde donde lo obtendrá el usuario y cual(es) rol(es) tendrá acceso		
25		Si es un reporte automático, se indica la ruta, frecuencia y usuario a ejecutarlo		
26		Para el reporte se requiere modificar/actualizar el perfil o perfiles de los usuarios		
27		Se indican campos requeridos por auditoria que debe mostrarse en el reporte		

9 <---- Cantidad de criterios evaluados

0 Cumplimientos

Puede utilizar este Checklist para realizar una validación de HU.

No realice modificaciones a esta lista; diríjalas al dpto de Calidad para su evaluación.

Este documento debe utilizarse en forma digital y no impresa.

Para ver los items que aplican en la revisión:
Para ver una lista de elementos marcados, seleccione **a** en el menú desplegable de la columna B celda #4.

Para volver a ver una lista de todas las tareas, seleccione **(Todas)** en el menú desplegable.

***** La nota mínima para aceptar una HU es 90 *****

Si alguna de las validaciones aplica y no es aprobada favor ingresar el detalle en la columna "Descripción", luego retornar al BA para su corrección; recuerde que debe abrir un bug referente al HU y a la carencia identificada, asignada al BA responsable.

Apéndice No 6. Matriz de dependencias

Nombre de la HU o dependencia	Area o a quién pertenece la HU o dependencia	Area o quién gestiona la HU o dependencia	Persona que soluciona la HU o la dependencia	Impacto	Fecha posible solución	Dependencias con otras HU	Descripción técnica

Apéndice No 7. *Sprint review*

Datos Generales

Scrum Master:	
Fecha:	
Participantes:	
Sprint:	
Proyecto:	

Estado del Sprint Backlog

ID Historia Usuario	Estado	Observaciones

Aprobaciones

Product Owner	
Firma:	
Observación:	

Apéndice No 8. *Sprint retrospective*

Datos Generales

Scrum Master:	
Fecha:	
Participantes:	
Sprint:	
Proyecto:	

Información de Relevante

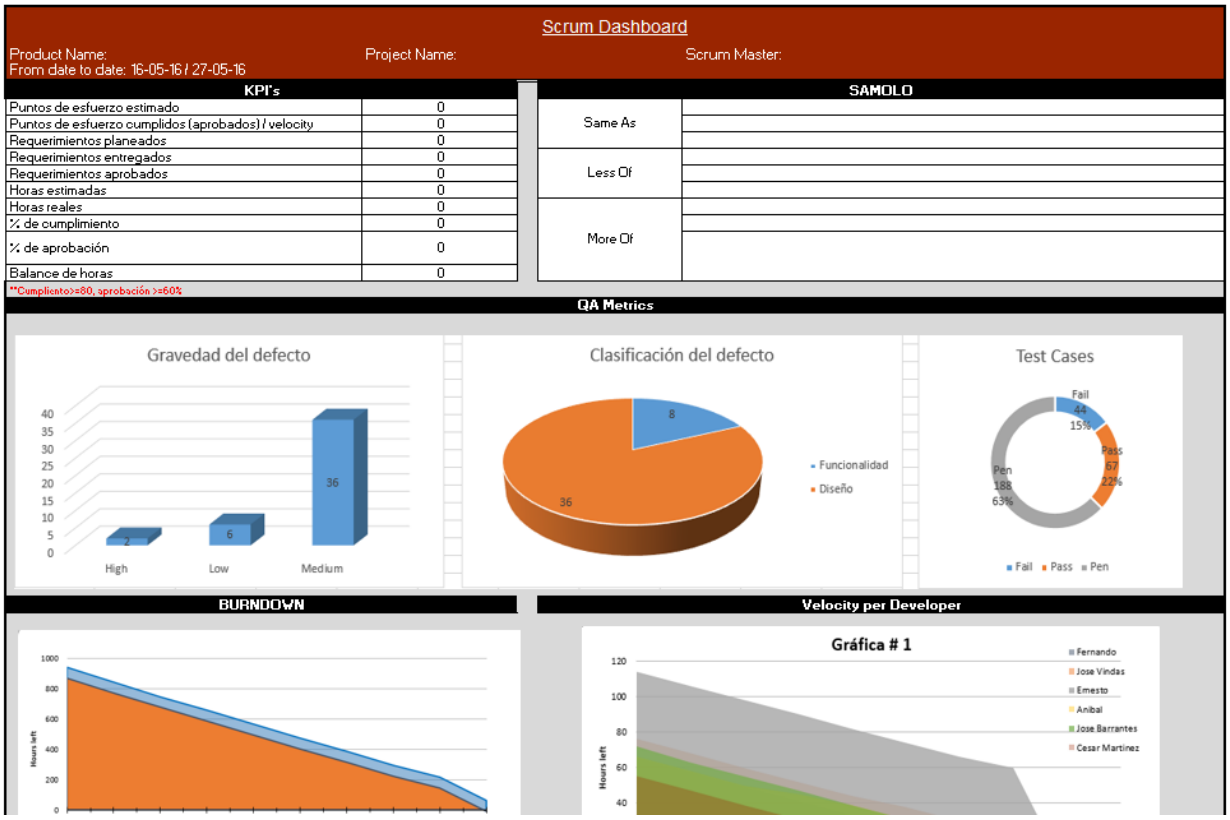
Explicación de dinámica:

¿Qué salió bien en la iteración? (aciertos)	¿Qué no salió bien en la iteración? (errores)	¿Qué mejoras vamos a implementar en la próxima iteración? (recomendaciones de mejora continua)

Nota:

- Se recomienda utilizar viñetas (bullets) para enumerar los aciertos, errores y recomendaciones de mejora continua.
- El formulario se puede extender cuantas páginas sea necesario para registrar todos los aciertos, errores y recomendaciones.

Apéndice No 9. *Sprint dashboard*



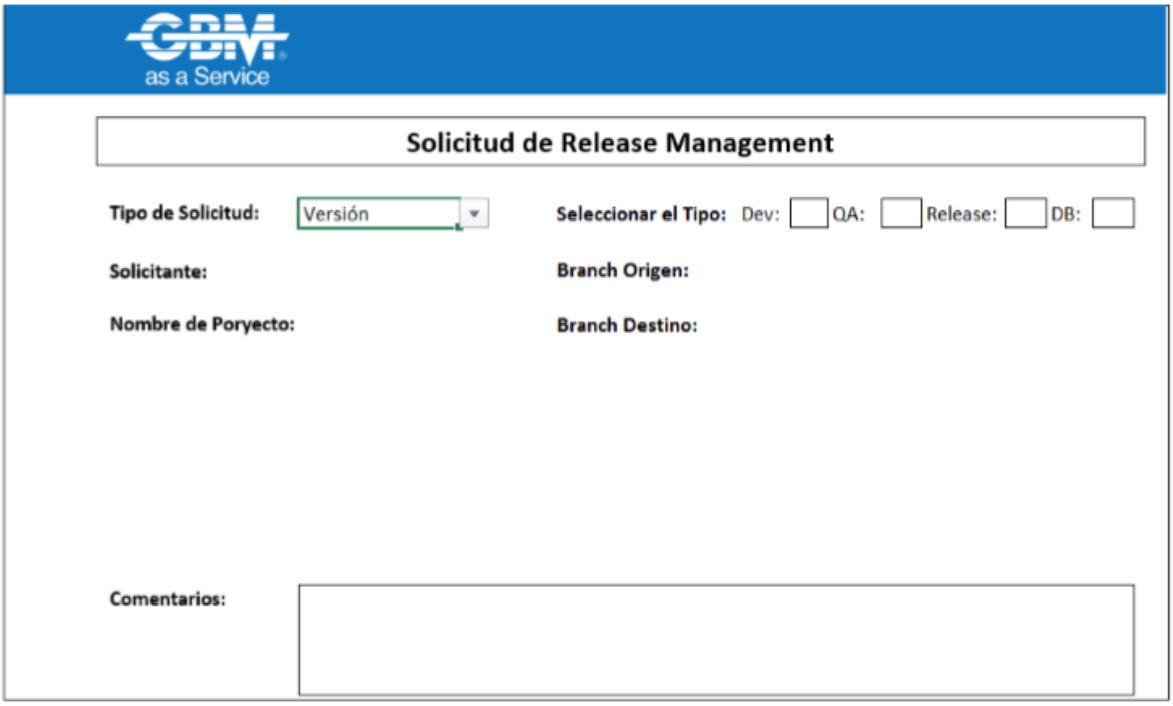
Apéndice No 10. Plan de *release*

Plan de Release Management

Plan de Release Management	
Nombre del Proyecto:	
Tech Lead: *Nombre del technical leader a cargo del proyecto*	
Scrum Master: *Nombre del Scrum Master establecido al proyecto*	
Tipo de Merge (Diario, Semanal): *El equipo debe de establecer cada cuanto van a realizar los merger*	
Día(s) para realizar el merge: *Establecer el día exacto de realizar los merge*	
Cantidad de Release que se requieren para el proyecto: *Establecer en número la cantidad de release que el proyecto requiera.*	
Establecer las fechas de esos release el tipo de release que se requiera.	
Fecha del Sprint de Release	**Tipo de Release (DEV, QA , Release Final)**

Apéndice No 11. Solicitud de *release*

Apéndice No 12. Solicitud de versionamiento



Apéndice No 13. ***Release notes***

Nombre del Proyecto: **Equipo:**				
Fecha	N° de Release	Funcionalidades Agregadas/Modificadas	Descripción	Bugs Cerrados

Apéndice No 14. Plan de pruebas

Tabla de contenido

1 Historial de Versiones 2

2 Información del Proyecto 2

3 Aprobaciones 2

4 Resumen Ejecutivo 3

5 Alcance de las Pruebas 3
- 5.1 Elementos de Pruebas 3
- 5.2 Nuevas Funcionalidades a Probar 3
- 5.3 Pruebas de Regresión 3
- 5.4 Funcionalidades a No Probar 3
- 5.5 Estrategia de Pruebas 3

6 Criterios de Aceptación o Rechazo 4
- 6.1 Criterios de Aceptación 4
- 6.2 Criterios de Suspensión 4
- 6.3 Criterios de Reanudación 4

7 Entregables 4

8 Recursos 4
- 8.1 Requerimientos de Entornos – Hardware 4
- 8.2 Requerimientos de Entornos – Software 4
- 8.3 Herramientas de Pruebas Requeridas 4
- 8.4 Colaboradores 4
- 8.5 Entrenamiento 4

9 Planificación y Organización 4
- 9.1 Definición de Escenarios y Casos de Prueba 4
- 9.2 Cronograma 5
- 9.3 Dependencias y Riesgos 5

Apéndice No 15. Análisis comparativo de herramientas colaborativas

Herramientas colaborativas para la gestión de requerimientos

Análisis comparativo de herramientas					
Disciplina	**Tools**	**Features**	**Pros**	**Contras**	**Costos**
Gestión de RQ's	**Door Next Generation**	1.Vínculo con Jira mediante Rational Adapter (véase análisis de brecha). 2. Herramienta orientada para organizaciones grandes que necesiten control	1.Analiza y gestiona cambios en los requisitos. 2. Requerimientos poseen ID. 3. Tipo de licenciamiento: onCloud, onPremise. 4. Conexión nativa con los productos IBM workbench.	1.Mayor costo por usuario. 2.Mayor complejidad de uso. 3.Integración con Jira mediante plug in. Véase documentos anexos	**On cloud:** **$164** usuario/mes con la herramienta Doors Next Generation. **On premise:** IBM Rational Doors Part number D0C29LL Part description IBM Rational DOORS Next Generation Analyst Floating User Single Install license + SW Subscription & Support 12 Months Charge unit Floating User Single Install **USD 11.600** IBM Rational Lifecycle Integration Adapters Tasktop Edition Part number D0UYDLL Part description IBM Rational Lifecycle Integration Adapters Tasktop Edition **Processor Value Unit (PVU)** License + SW Subscription & Support 12 Months **USD 592,00**
	Confluence	1.Vinculo Automático con Jira. 2.Control de versiones de documentos. 3.Documentos centralizados.	1.Visibilidad de los estados de las historias de usuario. 2.Permite gestión de documentación de proyectos ágiles. 3.Menor costo que la herammienta Doors Next Generation. 4.Trabajo colaborativo. 5.Tipo de licenciamiento: onCloud, onPremise.	1.No hay generación de ID de RQ, el ID es el generado de Jira y sólo se enlaza. 2.La visibilidad de los estados de las historias de usuario sólo se puede tener si tiene un usuario habilitado en confluence y además en jira; esto implicaría brindarle acceso a Jira al cliente.	**On cloud:** Precio usuarios/año: - 1-10=**$100** - 11-15=**$750** - 16-25=**$1.250** - 26-50=**$2.500** - 51-100=**$5.000** - 101-200=**$8.000**

Herramientas colaborativas para la gestión del proyecto en desarrollo

Análisis comparativo de herramientas					
Disciplina	**Tools**	**Features**	**Pros**	**Contras**	**Costos**
Gestión de Defectos	**IBM Rational Team Concert**	1.Herramienta para la gestión de planes, tareas, estados del proyecto y gestión de defectos; se integra de forma nativa con Rational Quality Manager para tener una trazabilidad entre los casos de prueba y los defectos registrados en la gestión de las pruebas.	1.Facilidad de aceso a la herramienta mediante la web. 2.Personalización de plantillas por medio de interfaz gráfica amigable. 3.Integración nativa con Rational Quality Manager para la relación de defectos con casos de prueba.	1. Generación de defectos poco intuitivo y es requerido realizar pasos adicionales en comparación con Jira. 2. Integración con herramientas de terceros es compleja, delicada y requiere costos adicionales. 3. En el laboratorio realizado se estima que ingresar un defecto con RTC podria tardar un 75% de tiempo adicional en comparación con Jira. 4. Duplicidad de información debido a que es necesario crear un proyecto en RTC y además en QM.	**On premise:** IBM Rational Team Concert Part number D0GNDLL Part description IBM Rational Team Concert **Contributor Authorized User** Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 1.910** Part number D0GMULL Part description IBM Rational Team Concert **Contributor Floating User** Single Install License + SW Subscription & Support 12 Months Charge unit Floating User Single Install **USD 5.710** Part number D0GMDLL Part description IBM Rational Team Concert **Developer Authorized User** Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 5.510**
Gestión del Proyecto	**Jira**	1.Herramienta en línea para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos ágiles. 2.Contiene informes incorporados de serie que recopilan datos prácticos y funcionales en tiempo real sobre el rendimiento del equipo sprint a sprint.	1.Capacidad de personalizar (formularios, flujos de trabajo, etc.) 2.La capacidad de compartir información fácilmente (agregar archivos adjuntos, agregar "observadores" a problemas, así como integraciones con otros productos). 3.Se adapta a la metodología ágil y es muy fácil de usar. 4.La interfaz es rápida y agradable de usar.	1.La capacidad de compartir flujos de trabajo entre proyectos se vuelve realmente complicada cuando se quiere cambiar uno de los proyectos sin cambiar el otro. 2.Flexibilidad limitada con los flujos de trabajo. 3.Carece de algunos informes específicos y de algunas capacidades de gráficos.	**On cloud:** 1.Hasta 10 usuarios $10 por usuario al mes **($1.200 por 10 usuarios al año)**. 2.De 11 a 100 usuarios $7 por usuario al mes **($8.400 por 100 usuarios año)**.

Herramientas colaborativas para la gestión de pruebas

Análisis comparativo de herramientas					
Disciplina	**Tools**	**Features**	**Pros**	**Contras**	**Costos**
Testing Funcional	IBM Quality Manager	1.Permite la integración con Funtional Tester para la grabación de los casos de prueba. 2.Facilidad de acceso vía Web a la herramienta.	1.Permite crear o modificar plantillas y flujos de trabajo según la necesidad. 2.Permite manejar flujos de aprobaciones brindando así trazabilidad de manera automatizada. 3.Permite reutilizar los casos de prueba para evitar duplicidad. 4.Cuenta con un historial de cada tarea, asignación o cambio realizado en la herramienta.	1.Herramienta de alto costo de licenciamiento en comparación con la herramienta PractiTest. 2.No se puede controlar acciones de edición de flujos; si se brinda accesos a un elemento el rol siempre tiene permiso para editar. 3.Reportería y dashboards son difíciles de obtener.	**On premise:** IBM Quality Manager Part number D05T9LL Part description IBM Rational Quality Manager **Contributor Authorized User** Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 1.910** Part number D05T5LL Part description IBM Rational Quality Manager Quality **Professional Authorized User** Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 6.710**
	IBM Funtional Tester	1.Herramienta para la creación y ejecución de pruebas funcionales automatizadas, utilizando el reconocimiento de objetos para la creación de los scripts permitiendo generar pruebas más fiables y de fácil mantenimiento.	1. Amplio catálogo de técnologías soportadas. 2. Facilidades amplias para la generación y mantenibilidad de scripts. 3. Automatización de pruebas funcionales para ambientes web y de escritorio. 4.Variedad de formatos para la generación de reportes.	1.Soporte limitado y restricción de navegadores web sobre los cuáles se pueden aplicar pruebas. 2.Soporte limitado y uso no recomendado sobre aplicaciones móviles. 3.No compatible para realizar pruebas sobre integraciones.	**On premise:** IBM Functional Tester Part number D53NFLL Part description IBM Rational Functional **Tester Authorized User** License + SW Subscription & Support 12 Months Charge unit Authorized User **USD 7.670** Part number D530BLL Part description IBM Rational Functional **Tester Floating User** License + SW Subscription & Support 12 Months Charge unit Floating User **USD 14.700**
	Katalon	1.Herramienta para la creación y ejecución de pruebas funcionales automatizadas, incluyendo web, mobile y API's.	1. Amplio catálogo de técnologías soportadas. 2. Soporta distintas plataformas (web, mobile, api's). 3. Integración con soluciones SLDC (Jenkins, Git, Jira, qTest). 4.Es más amigable de utilizar.	1.En algunas ocasiones es lento. 2.Es necesario reinstalarlo para actualizarlo.	**On premise:** Katalon Studio gratis.
	Practitest	1.Herramienta para la gestión de pruebas. 2.Apta para el manejo de problemas y seguimiento de errores. 3.Puede tener toda la información (ejecuciones, fallos, creación) en una solución integrada de gestión de casos de prueba. 4.Permite la trazabilidad completa, desde defectos hasta pruebas y requisitos. 5.Tiene la capacidad para administrar pruebas manuales y automáticas.	1.Permite crear campos y filtros personalizados. 2.Interfaz gráfica de usuario fácil de utilizar, vía web. 3.Permite generar dashboard's e informes de las pruebas. 4.Permite la Integración entre requisitos, casos de prueba, ejecución y defectos con herramientas como: JIRA, Pivotal Tracker, Jenkins, etc. 5.Permite almacenar base de conocimiento y videos relacionados a evidencias. 6.Permite el control de acceso a los usuarios mediante grupos y permisos específicos. 7.El licenciamiento es bajo respecto a herramientas como IBM Rational Quality Manager.	1.Sincronización con herramientas de terceros limitada a los atributos predefinidos por practiTest.	**On cloud:** 1.Licencia Profesional: (Se deben adquirir mínimo 3 licencias de pruebas) **$35 por usuario por mes ($1.260 por año)**. 2.Licencia Enterprice: (Se deben adquirir mínimo 3 licencias de pruebas) **$45 por usuario por mes ($1.620 por año)**. Referencia: https://www.practitest.com/pricing/

Análisis comparativo de herramientas

Disciplina	Tools	Features	Pros	Contras	Costos
Performance Testing	IBM Performance Tester	1.Valida la escalabilidad de las aplicaciones web y de servidor.	1.Buen rendimiento a la hora de ejecutar las pruebas. 2.Reportes muy completos y con posibilidad de personalizarlos. 3.Fácil configuración de los datapool y manejo por medio de importación y exportación de éstos. 4.Se puede utilizar hasta 5 usuarios múltiples por ejecución, para la version trial. 5.Permite realizar pruebas sin necesidad de incluir código. 6.Permite incluir más funcionalidad de una prueba por medio de código en Java. 7.Cuenta con informes en tiempo real, logrando identificar de manera inmediata los problemas en un navegador de resultados vía web. 8.Posibilidad de hacer ejecuciones múltiples por medio de Schedules. 9.Se pueden hacer puntos de verificación por medio de código en Java. 10.Para pruebas, según su demanda, se pueden adquirir usuarios adicionales.	1.No es factible utilizar esta herramienta en dispositivos moviles. 2. El licenciamiento de usuarios virtuales es alto (ver siguiente columna).	**On premise:** IBM Performance Tester Part number D54LRLL Part description IBM Rational Performance **Tester Authorized User** License + SW Subscription & Support 12 Months Charge unit Authorized User **USD 2.250** Part number D54LTLL Part description IBM Rational Performance **Tester Floating User** License + SW Subscription & Support 12 Months Charge unit Floating User **USD 4.480** **Virtual Users** Part number D530ALL Part description IBM Rational Performance **Test Pack 250 Virtual Testers Floating** User License + SW Subscription & Support 12 Months **USD 48,700.00** Part number D530ILL Part description IBM Rational Performance **Test Pack Virtual Testers 500 Floating** User License + SW Subscription & Support 12 Months **USD 69,100.00** D1BF8LL IBM Rational Performance **Test Pack Virtual Testers 250 Floating** Users Monthly License **Month USD 2.030** D1BF9LL IBM Rational Performance **Test Pack Virtual Testers 500 Floating** User Monthly License
	Jmeter	1.Software de código abierto. 2.Aplicación Java 100%. 3.Diseñada para cargar el comportamiento funcional de la prueba y medir el rendimiento. 4.Capacidad para cargar y probar el rendimiento de diferentes aplicaciones y tipos de protocolos web. 5.Originalmente fue diseñado para probar aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba.	1.Cuenta con soporte mediante la comunidad web. 2.Se puede integrar con Selenium, Eclipse, Jenkins, TeamCity para desarrollar pruebas efectivas en la capa de interfaz de usuario. 3.Permite extraer datos y graficarlos sin tener que diseñarlos. 4.Es compatible con cualquier lenguaje de código abierto como Java, Javascript, BeanShell, Groovy.	1.Al diseñar pruebas de API con JMeter, la espera de subprocesos es muy limitada. 2.La grabación de pruebas es muy compleja para un usuario con menos habilidades de programación. 3.Al ser de código abierto, podría tener errores. 4.El uso de la herramienta aumenta el consumo local de RAM. 5.Ayuda con alcance limitado ya que es una comunidad.	**On premise:** Jmeter gratis.
Integration y Virtualization	IBM Rational Integration Tester	1.Compatible con diferentes protocolos de integración. 2.Ejecución de pruebas en diferentes formatos de Web Services. 3.Integración con IBM Rational Quality Manager (adaptador). 4.Permite realizar pruebas de regresión. 5.Creación automática de reportes de los resultados de las pruebas.	No se logró obtener información sobre este punto en la web y recursos en línea.	No se logró obtener información sobre este punto en la web y recursos en línea.	**On cloud:** Debe adquirirse Test Workbench; se muestra licenciamiento. *Licencia 12 meses:* Part number D0PRPLL Part description IBM Rational **Test Workbench Floating** User Single Install License + SW Subscription & Support 12 Months **USD 26.000** Licencia Monthly: D1BMILL IBM Rational **Test Workbench Floating** User Single Install Monthly License **USD 1.080**
	IBM Rational Test Virtualization Server	1.Virtualización de servicios, software y aplicaciones. 2.Actualización, reutilización y ambientes virtuales compartidos. 3.Integración con otras herramientas de Rational.	No se logró obtener información sobre este punto en la web y recursos en línea.	No se logró obtener información sobre este punto en la web y recursos en línea.	**On premise:** IBM Rational Test Virtualization Server Part number D0PTZLL Part description IBM **Rational Test Virtualization Server Processor Value Unit** (PVU) License + SW Subscription & Support 12 Months Charge unit Processor Value Unit (PVU) **USD 2.630**
Security Testing	Appscan	1.Escaner de tipo caja blanca. 2.Escanea vulnearabilidades potenciales en código fuente. 3.Compatibilidad con herramientas y tecnologías conocidas de desarrollo.	1.Posibilidad de encontrar vulnerabilidades previo al lanzamiento a producción. 2.Compatibilidad con gran número lenguajes de desarrollo comúnmente utilizados	1.Herramienta de alto costo de licenciamiento (ver columna de costo).	**On premise:** Precio USD 29.000 por una licencia perpetua.

Apéndice No 16. *Peer review*

Información del sistema			
Sistema (s):			
Número de necesidad de servicio:			
Nombre del revisor			
Descripción del ajuste:			
Nombre del desarrollador:		**Tipo de desarrollador:** <Marque con X> _____ **Interno** <Marque con X> _____ **Externo**	
Fecha de revisión:			
Revisión de código fuente			
1. Aplica la revisión del código fuente:			
Si	**No**	**Criterio de evaluación**	
		Es un programa sensible que afecta saldos de clientes.	
		Es un programa sensible que elimina o actualiza registrado.	
		Es un programa sensible que genera transacciones.	
		Es un programa sensible que afecta directamente el servicio al cliente externo.	
		Otros elementos considerados válidos para asignar la revisión del código: <Indicar cuales>	
2. Se identificó código malintencionado:			
Si	**No**	**Observaciones**	
3. Se usaron métodos de acceso correctos a la base de datos para el cambio aplicado:			
Si	**No**	**Observaciones**	
4. Se identificaron datos de prueba en el código fuente aplicado:			
Si	**No**	**Observaciones**	
5. Los parámetros cumplen con lo establecido:			
Si	**No**	**Observaciones**	
6. Se identificó código no alineado al estándar:			
Si	**No**	**Observaciones**	
7. Se identificó código no eficiente:			
Si	**No**	**Observaciones**	
8. Se avala el desarrollo realizado para este ajuste:			
Si	**No**	**Observaciones**	
		<En caso de que aplique, indicar observaciones>	

Análisis comparativo de herramientas colaborativas					
Disciplina	***Tools***	***Features***	**Pros**	**Contras**	**Costos**
Gestión de RQ	**Door Next Generation**	1.Vínculo con Jira mediante Rational Adapter (véase análisis de brecha). 2. Herramienta orientada para organizaciones grandes que necesiten control.	1.Analiza y gestiona cambios en los requisitos. 2. Requerimientos poseen ID. 3. Tipo de licenciamiento: onCloud, onPremise. 4. Conexión nativa con los productos IBM workbench.	1.Mayor costo por usuario. 2.Mayor complejidad de uso. 3.Integración con Jira mediante *plug in*. Véase documentos anexos.	**<u>On cloud:</u>** **$164** usuario/mes con la herramienta Doors Next Generation. **<u>On premise:</u>** IBM Rational Doors Part number D0C29LL Part description IBM Rational DOORS Next Generation Analyst Floating User Single Install license + SW Subscription & Support 12 Months Charge unit Floating User Single Install **USD 11.600** IBM Rational Lifecycle Integration Adapters Tasktop Edition Part number D0UYDLL Part description IBM Rational Lifecycle Integration Adapters Tasktop Edition **Processor Value Unit (PVU)** License + SW Subscription & Support 12 Months **USD 592,00**

	Confluence	1.Vínculo automático con Jira. 2.Control de versiones de documentos. 3.Documentos centralizados.	1.Visibilidad de los estados de las historias de usuario. 2.Permite gestión de documentación de proyectos ágiles. 3.Menor costo que la herammienta Doors Next Generation. 4.Trabajo colaborativo. 5.Tipo de licenciamiento: onCloud, onPremise.	1.No hay generación de ID de RQ, el ID es el generado de Jira y solo se enlaza. 2.La visibilidad de los estados de las historias de usuario solo se puede tener si tiene un usuario habilitado en Confluence y, además, en jira; esto implicaría brindarle acceso a Jira al cliente.	**<u>On cloud:</u>** Precio usuarios/año: - 1-10=**$100** - 11-15=**$750** - 16-25=**$1.250** - 26-50=**$2.500** - 51-100=**$5.000** - 101-200=**$8.000**
Testing Funcional	**IBM Quality Manager**	1.Permite la integración con Funtional Tester para la grabación de los casos de prueba. 2.Facilidad de acceso vía web a la herramienta.	1.Permite crear o modificar plantillas y flujos de trabajo según la necesidad. 2.Permite manejar flujos de aprobaciones, brindando así trazabilidad de manera automatizada. 3.Permite reutilizar los casos de prueba para evitar duplicidad.	1.Herramienta de alto costo de licenciamiento en comparación con la herramienta PractiTest. 2.No se pueden controlar acciones de edición de flujos; si se brinda accesos a un elemento, el rol siempre tiene permiso para editar. 3.Reportería y	**<u>On premise:</u>** IBM Quality Manager Part number D05T9LL Part description IBM Rational Quality Manager **Contributor Authorized User** Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 1.910** Part number D05T5LL Part description IBM Rational Quality Manager Quality **Professional Authorized User** Single Install License + SW Subscription & Support

			4.Cuenta con un historial de cada tarea, asignación o cambio realizado en la herramienta.	*dashboards* son difíciles de obtener.	12 Months Charge unit Authorized User Single Install **USD 6.710**
	IBM Funtional Tester	1.Herramienta para la creación y ejecución de pruebas funcionales automatizadas, utilizando el reconocimiento de objetos para la creación de los *scripts* permitiendo generar pruebas más fiables y de fácil mantenimiento.	1. Amplio catálogo de tecnologías soportadas. 2. Facilidades amplias para la generación y mantenibilidad de *scripts.* 3. Automatización de pruebas funcionales para ambientes web y de escritorio. 4. Variedad de formatos para la generación de reportes.	1.Soporte limitado y restricción de navegadores web sobre los cuales se pueden aplicar pruebas. 2. Soporte limitado y uso no recomendado sobre aplicaciones móviles. 3. No compatible para realizar pruebas sobre integraciones.	**<u>On premise:</u>** IBM Functional Tester Part number D53NFLL Part description IBM Rational Functional **Tester Authorized User** License + SW Subscription & Support 12 Months Charge unit Authorized User **USD 7.670** Part number D530BLL Part description IBM Rational Functional **Tester Floating User** License + SW Subscription & Support 12 Months Charge unit Floating User **USD 14.700**

	Katalon	1.Herramienta para la creación y ejecución de pruebas funcionales automatizadas, incluyendo web, *mobile* y API.	1. Amplio catálogo de tecnologías soportadas. 2. Soporta distintas plataformas (web, mobile, API). 3. Integración con soluciones SLDC (Jenkins, Git, Jira, qTest). 4.Es más amigable de utilizar.	1.En algunas ocasiones es lento. 2. Es necesario reinstalarlo para actualizarlo.	**<u>On premise:</u>** Katalon Studio gratis.
	Practitest	1.Herramienta para la gestión de pruebas. 2.Apta para el manejo de problemas y seguimiento de errores. 3.Puede tener toda la información (ejecuciones, fallos, creación) en una solución integrada de gestión de casos de prueba. 4.Permite la trazabilidad completa, desde defectos hasta pruebas y requisitos.	1.Permite crear campos y filtros personalizados. 2.Interfaz gráfica de usuario fácil de utilizar, vía web. 3.Permite generar *dashboards* e informes de las pruebas. 4.Permite la integración entre requisitos, casos de prueba, ejecución y defectos con herramientas como JIRA, Pivotal Tracker, Jenkins, etc. 5.Permite almacenar base de	1.Sincronización con herramientas de terceros limitada a los atributos predefinidos por practiTest.	**<u>On cloud:</u>** 1.Licencia profesional: (Se deben adquirir mínimo 3 licencias de pruebas) **$35 por usuario por mes ($1.260 por año)**. 2.Licencia Enterprice: (Se deben adquirir mínimo 3 licencias de pruebas) **$45 por usuario por mes ($1.620 por año)**. Referencia: https://www.practitest.com/pricing/

		5.Tiene la capacidad para administrar pruebas manuales y automáticas.	conocimiento y videos relacionados con evidencias. 6.Permite el control de acceso a los usuarios mediante grupos y permisos específicos. 7.El licenciamiento es bajo, respecto a herramientas como IBM Rational Quality Manager.		

Performance Testing	**IBM Performance Tester**	1.Valida la escalabilidad de las aplicaciones web y de servidor.	1.Buen rendimiento a la hora de ejecutar las pruebas. 2.Reportes muy completos y con posibilidad de personalizarlos. 3.Fácil configuración de los *datapool* y manejo por medio de importación y exportación de estos. 4.Se pueden utilizar hasta cinco usuarios múltiples por ejecución, para la versión trial. 5.Permite realizar pruebas sin necesidad de incluir código. 6.Permite incluir más funcionalidad de una prueba por medio de código en Java. 7.Cuenta con informes en tiempo real, logrando identificar de	1.No es factible utilizar esta herramienta en dispositivos móviles. 2. El licenciamiento de usuarios virtuales es alto (ver siguiente columna).	**<u>On premise:</u>** IBM Performance Tester Part number D54LRLL Part description IBM Rational Performance **Tester Authorized User** License + SW Subscription & Support 12 Months Charge unit Authorized User **USD 2.250** Part number D54LTLL Part description IBM Rational Performance **Tester Floating User** License + SW Subscription & Support 12 Months Charge unit Floating User **USD 4.480** **<u>Virtual Users</u>** Part number D530ALL Part description IBM Rational Performance **Test Pack 250 Virtual Testers Floating** User License + SW Subscription & Support 12 Months **USD 48,700.00** Part number D530ILL Part description IBM Rational Performance **Test Pack Virtual Testers 500 Floating** User License + SW Subscription & Support 12 Months **USD 69,100.00**

			manera inmediata los problemas en un navegador de resultados vía web. 8.Posibilidad de hacer ejecuciones múltiples por medio de *schedules.* 9.Se pueden hacer puntos de verificación por medio de código en Java. 10.Para pruebas, según su demanda, se pueden adquirir usuarios adicionales.		D1BF8LL IBM Rational Performance **Test Pack Virtual Testers 250 Floating** Users Monthly License **Month USD 2.030** D1BF9LL IBM Rational Performance **Test Pack Virtual Testers 500 Floating** User Monthly License **Month USD 2.870**

	Jmeter	1.*Software* de código abierto. 2.Aplicación Java 100%. 3.Diseñada para cargar el comportamiento funcional de la prueba y medir el rendimiento. 4.Capacidad para cargar y probar el rendimiento de diferentes aplicaciones y tipos de protocolos web. 5.Originalmente fue diseñado para probar aplicaciones web, pero desde entonces se ha expandido a otras funciones de prueba.	1.Cuenta con soporte mediante la comunidad web. 2.Se puede integrar con Selenium, Eclipse, Jenkins, TeamCity para desarrollar pruebas efectivas en la capa de interfaz de usuario. 3.Permite extraer datos y graficarlos sin tener que diseñarlos. 4.Es compatible con cualquier lenguaje de código abierto como Java, Javascript, BeanShell, Groovy.	1.Al diseñar pruebas de API con JMeter, la espera de subprocesos es muy limitada. 2.La grabación de pruebas es muy compleja para un usuario con menos habilidades de programación. 3.Al ser de código abierto, podría tener errores. 4.El uso de la herramienta aumenta el consumo local de RAM. 5.Ayuda con alcance limitado, ya que es una comunidad.	**<u>On premise:</u>** Jmeter gratis.

Integration y Virtualization	**IBM Rational Integration Tester**	1.Compatible con diferentes protocolos de integración. 2. Ejecución de pruebas en diferentes formatos de Web Services. 3. Integración con IBM Rational Quality Manager (adaptador). 4.Permite realizar pruebas de regresión. 5.Creación automática de reportes de los resultados de las pruebas.	No se logró obtener información sobre este punto en la web y recursos en línea.	No se logró obtener información sobre este punto en la web y recursos en línea.	**<u>On cloud:</u>** Debe adquirirse Test Workbench; se muestra licenciamiento. *<u>Licencia 12 meses:</u>* Part number D0PRPLL Part description IBM Rational **Test Workbench Floating** User Single Install License + SW Subscription & Support 12 Months **USD 26.000** Licencia Monthly: D1BMILL IBM Rational **Test Workbench Floating** User Single Install Monthly License **USD 1.080**
	IBM Rational Test Virtualization Server	1.Virtualización de servicios, *software* y aplicaciones. 2. Actualización, reutilización y ambientes virtuales compartidos. 3. Integración con otras herramientas de Rational.	No se logró obtener información sobre este punto en la web y recursos en línea.	No se logró obtener información sobre este punto en la web y recursos en línea.	**<u>On premise:</u>** IBM Rational Test Virtualization Server Part number D0PTZLL Part description IBM **Rational Test Virtualization Server Processor Value Unit** (PVU) License + SW Subscription & Support 12 Months Charge unit Processor Value Unit (PVU) **USD 2.630**

Security Testing	**Appscan**	1.Escáner de tipo caja blanca. 2. Escanea vulnerabilidades potenciales en código fuente. 3.Compatibilidad con herramientas y tecnologías conocidas de desarrollo.	1.Posibilidad de encontrar vulnerabilidades previo al lanzamiento a producción. 2.Compatibilidad con gran número lenguajes de desarrollo comúnmente utilizados	1.Herramienta de alto costo de licenciamiento (ver columna de costo).	**<u>On premise:</u>** Precio USD 29.000 por una licencia perpetua.
Gestión de Defectos	**IBM Rational Team Concert**	1.Herramienta para la gestión de planes, tareas, estados del proyecto y gestión de defectos; se integra de forma nativa con Rational Quality Manager para tener una trazabilidad entre los casos de prueba y los defectos registrados en la gestión de las pruebas.	1.Facilidad de acceso a la herramienta mediante la web. 2.Personalización de plantillas por medio de interfaz gráfica amigable. 3.Integración nativa con Rational Quality Manager para la relación de defectos con casos de prueba.	1. Generación de defectos poco intuitiva y es requerido realizar pasos adicionales en comparación con Jira. 2. Integración con herramientas de terceros es compleja, delicada y requiere costos adicionales. 3. En el laboratorio realizado, se estima que ingresar un defecto con RTC podría tardar un 75% de tiempo	**<u>On premise:</u>** IBM Rational Team Concert Part number D0GNDLL Part description IBM Rational Team Concert **Contributor Authorized User** Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 1.910** Part number D0GMULL Part description IBM Rational Team Concert **Contributor Floating User** Single Install License + SW Subscription & Support 12 Months Charge unit Floating User Single Install **USD 5.710** Part number D0GMDLL Part description IBM Rational Team Concert **Developer Authorized User**

				adicional en comparación con Jira. 4. Duplicidad de información, debido a que es necesario crear un proyecto en RTC y, además, en QM.	Single Install License + SW Subscription & Support 12 Months Charge unit Authorized User Single Install **USD 5.510**
Gestión del Proyecto	**Jira**	1.Herramienta en línea para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos ágiles. 2.Contiene informes incorporados de	1.Capacidad de personalizar (formularios, flujos de trabajo, etc.). 2.La capacidad de compartir información fácilmente (agregar archivos adjuntos, agregar "observadores" a problemas, así como	1.La capacidad de compartir flujos de trabajo entre proyectos se vuelve realmente complicada cuando se quiere cambiar uno de los proyectos sin cambiar el otro. 2.Flexibilidad limitada con los flujos de trabajo. 3.Carece de	**<u>On cloud:</u>** 1.Hasta 10 usuarios, $10 por usuario al mes **($1.200 por 10 usuarios al año)**. 2.De 11 a 100 usuarios, $7 por usuario al mes **($8.400 por 100 usuarios año)**.

		serie que recopilan datos prácticos y funcionales en tiempo real sobre el rendimiento del equipo *sprint* a *sprint*.	integraciones con otros productos). 3.Se adapta a la metodología ágil y es muy fácil de usar. 4.La interfaz es rápida y agradable de usar.	algunos informes específicos y de algunas capacidades de gráficos.	

Apéndice No 17. Plan de implementación

Nombre de tarea	Duración	Trabajo	Comienzo	Fin	Predecesoras	Nombres de los recursos
Plan piloto fábrica de Integration	**89,5 días**	**6 538 horas**	**mié 2/5/18**	**mar 4/9/18**		
Diseño de propuesta	**50 días**	**474 horas**	**mié 2/5/18**	**mar 10/7/18**		
Planeación, Marco teórico	74 horas	74 horas				Consultor
Recolección de información	160 horas	160 horas	mié 2/5/18	mar 29/5/18		Consultor
Diagnóstico	80 horas	80 horas	mié 30/5/18	mar 12/6/18	4	Consultor
Propuesta	160 horas	160 horas	mié 13/6/18	mar 10/7/18	5	Consultor
Planeación	**3 días**	**72 horas**	**mié 2/5/18**	**vie 4/5/18**		
Definición de roles dentro del equipo	8 horas	8 horas	mié 2/5/18	mié 2/5/18		Scrum Coach
Configuración de las herramientas colaborativas	16 horas	64 horas	jue 3/5/18	vie 4/5/18	8	Scrum Coach; Requirements Coach; QA Coach; Release management Coach
Preparación para capacitación	**24,5 días**	**128 horas**	**lun 7/5/18**	**vie 8/6/18**		
Creación de presentaciones en Power Point para talleres	4 horas	16 horas	lun 7/5/18	lun 7/5/18	9	QA Coach; Release management Coach; Requirements Coach; Scrum Coach
Creación de *E-Learning* de cada práctica ágil	16 horas	64 horas	lun 7/5/18	jue 10/5/18	11	QA Coach; Release management Coach; Requirements Coach; Scrum Coach
Creación de *brochures* material	8 horas	32 horas	jue 7/6/18	jue 7/6/18	12	QA Coach; Release management Coach; Requirements Coach; Scrum Coach

gestión de cambio						
Crear exámenes para *e-learning*	4 horas	16 horas	vie 8/6/18	vie 8/6/18	13	QA Coach; Release management Coach; Requirements Coach; Scrum Coach
Ejecución de capacitaciones	**1 día**	**96 horas**	**vie 8/6/18**	**lun 11/6/18**	**10**	
Capacitación metodológica	4 horas	48 horas	vie 8/6/18	vie 8/6/18		QA Coach[0,25];Release management Coach[0,25];Requirements Coach[0,25];Scrum Coach[0,25];Dev 1;Dev 2;Dev 3;Dev 4;Dev 5;Dev 6;Dev 7;Dev 8;Dev 9;Dev 10;Dev 11
Capacitación herramientas colaborativas	4 horas	48 horas	lun 11/6/18	lun 11/6/18	16	QA Coach[0,25];Release management Coach[0,25];Requirements Coach[0,25];Scrum Coach[0,25];Dev 1;Dev 2;Dev 3;Dev 4;Dev 5;Dev 6;Dev 7;Dev 8;Dev 9;Dev 10;Dev 11
Ejecución del piloto	**61 días**	**5 768 horas**	**lun 11/6/18**	**mar 4/9/18**		
Coaching	60 días	5 760 horas	lun 11/6/18	lun 3/9/18	17	QA Coach[0,25];Release management Coach[0,25];Requirements Coach[0,25];Scrum Coach[0,25];Dev 1;Dev 2;Dev 3;Dev 4;Dev 5;Dev 6;Dev 7;Dev 8;Dev 9;Dev 10;Dev 11
Ajustes	8 horas	8 horas	lun 3/9/18	mar 4/9/18	19	QA Coach[0,25];Release management Coach[0,25];Requirements Coach[0,25];Scrum Coach[0,25]

Nota: elaboración propia.

Apéndice No 18. Costo de mano de obra

Recurso	Cantidad de horas	Costo por Hora	Cantidad de recursos	Costo sin cargas Sociales	Porcentaje de cargas sociales	Total	Costo total
Consultor propuesta	474	**60**	**1**	28440	50,33	14313,85	42753,85
Scrum Coach	180	60	1	10800	50,33	5435,64	16235,64
Requirements Coach	172	60	1	10320	50,33	5194,06	15514,06
QA Coach	172	60	1	10320	50,33	5194,06	15514,06
Release Manager Coach	172	60	1	10320	50,33	5194,06	15514,06
Desarrolladores/QAs (capacitación y ejecución del piloto)	488	45	11	241560	50,33	121577,15	363137,15
Total	1658			311760	50,33	156908,81	468668,81

Nota: El empleo.com

Apéndice No 19. Costo de materiales de capacitación

Material	Cantidad	Detalle	Costo/unidad	Costo total
Papel de construcción	2	Block (rojo, rosado, amarillo, beige, café, blanco, celeste, azul)	3290	6580
Pliego papel periódico	11	Pliegos	300	3300
Goma	4	Pequeñas	1440	5760
Tijeras	4		1990	7960
Regla	4	30 cm	490	1960
Cartulina	1	Blanca	6290	6290
Marcadores	12	Negros	490	5880
Post its	5	Blocks multicolor	5990	29950
Marshmallows	2	Bolsa tamaño mediano de *marshmellow*	2000	4000
Spaguettis	2	Bolsa grande de *spaguettis* delgados	1250	2500
Masking tape	2	Delgado	840	1680
Suma total				75860
Suma total dólares		**Costo del dólar**	562,15	134,95

Nota: elaboración propia.